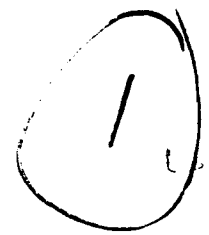


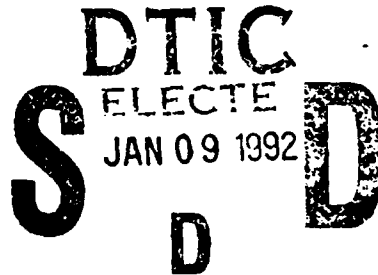
AD-A244 975



**SOFTWARE DESIGN DOCUMENT
SAF Simulation Host CSCI (8)**

Volume 1 of 2 Sections 1.0 - 2.7

June, 1991



Prepared by:

BBN Systems and Technologies,
A Division of Bolt Beranek and Newman Inc.
10 Moulton Street
Cambridge, MA 02138
(617) 873-3000 FAX: (617) 873-4315

Prepared for:

Defense Advanced Research Projects Agency (DARPA)
Information and Science Technology Office
1400 Wilson Blvd., Arlington, VA 22209-2308
(202) 694-8232, AUTOVON 224-8232

Program Manager for Training Devices (PM TRADE)
12350 Research Parkway
Orlando, FL 32826-3276
(407) 380-4518

92-00252



**APPROVED FOR PUBLIC RELEASE
DISTRIBUTION UNLIMITED**

92 1 6 060

REPORT DOCUMENTATION PAGE

Form Approved
OPM No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Information and Regulatory Affairs, Office of Management and Budget, Washington, DC 20503.

1. AGENCY USE ONLY (Leave Blank)		2. REPORT DATE June 1991		3. REPORT TYPE AND DATES COVERED Software Design Document	
4. TITLE AND SUBTITLE Software Design Document SAF Simulation Host CSCI (8)				5. FUNDING NUMBERS Contract Numbers: MDA972-89-C-0060 MDA972-89-C-0061	
6. AUTHOR(S) Author not specified.					
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Bolt Beranek and Newman, Inc. (BBN) Systems and Technologies; Advanced Simulation 10 Moulton Street Cambridge, MA 02138				8. PERFORMING ORGANIZATION REPORT NUMBER Advanced Simulation #: 9111	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Defense Advanced Research Projects Agency (DARPA) 3701 North Fairfax Drive Arlington, VA 22203-1714				10. SPONSORING/MONITORING AGENCY REPORT NUMBER DARPA Report Number: None.	
11. SUPPLEMENTARY NOTES None					
12a. DISTRIBUTION/AVAILABILITY STATEMENT Distribution Statement A: Approved for public release; distribution is unlimited.				12b. DISTRIBUTION CODE Distribution Code: A	
13. ABSTRACT (Maximum 200 words) A Simulation Network (SIMNET) project Software Design Document that describes the Semi-Automated Forces (SAF) Simulation Host Computer Software Configuration Item (CSCI number 8) of the SIMNET hardware and software training system for vehicle crew training and operational training.					
14. SUBJECT TERMS SIMNET Software Design Document for the SAF Simulation Host CSCI (CSCI 8).				15. NUMBER OF PAGES	
				16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT Same as report.		

SOFTWARE DESIGN DOCUMENT

SAF Simulation Host CSCI (8)

Volume 1 of 2 Sections 1.0 - 2.7

June, 1991

Prepared by:

BBN Systems and Technologies,
A Division of Bolt Beranek and Newman Inc.
10 Moulton Street
Cambridge, MA 02138
(617) 873-3000 FAX: (617) 873-4315

Prepared for:

Defense Advanced Research Projects Agency (DARPA)
Information and Science Technology Office
1400 Wilson Blvd., Arlington, VA 22209-2308
(202) 694-8232, AUTOVON 224-8232

Program Manager for Training Devices (PM TRADE)
12350 Research Parkway
Orlando, FL 32826-3276
(407) 380-4518



Accession For	
NTIS	GRAM
DIIC	IAJ
Use	on bid
Justification	
By	
Distribution	
Availability Codes	
Dist	Availability for Special
A-1	

APPROVED FOR PUBLIC RELEASE
DISTRIBUTION UNLIMITED

Table of Contents

1	INTRODUCTION : SAF SIMULATION HOST CSCI.....	1
1.1	BACKGROUND.....	1
1.2	EXTERNAL INTERFACES	1
1.3	INTERNAL STRUCTURE.....	2
1.4	CONFIGURATION AND CONFIGURATION MANAGEMENT	3
1.5	TERMINOLOGY AND DOCUMENTATION	4
2	CSC DESCRIPTIONS	5
2.1.	INITIALIZATION CSC.....	5
2.1.1	libreader CSC.....	6
2.1.1.1	parser.y CSC	6
2.1.1.1.1	stack_push CSU.....	7
2.1.1.1.2	stack_push_array CSU	7
2.1.1.1.3	free_stack CSU.....	7
2.1.1.1.4	make_array CSU	8
2.1.1.1.5	copy_stack_to_array CSU	8
2.1.1.1.6	yyerror CSU	9
2.1.1.1.7	reader_read_file CSU.....	9
2.1.1.2	lexer.l CSU	9
2.1.1.3	symbol.c CSC	9
2.1.1.3.1	init_symbol_table CSU.....	10
2.1.1.3.2	get_symbol CSU	11
2.1.1.3.3	get_string CSU	11
2.1.1.3.4	describe_symbol_table CSU.....	12
2.1.1.3.5	Do_Hash CSU.....	12
2.1.1.3.6	get_symbol_value CSU	12
2.1.1.3.7	set_symbol_value CSU.....	13
2.1.1.4	tags.c CSC.....	13
2.1.1.4.1	is_probably_a_string CSU.....	13
2.1.1.4.2	tag_error CSU	14
2.1.1.4.3	find_tag CSU	14
2.1.1.4.4	cmp_tags CSU.....	15
2.1.1.4.5	sort_tag_table CSU	15
2.1.1.4.6	binarysearch_tag CSU.....	16
2.1.1.4.7	find_tag_sorted CSU	16
2.1.1.5	libreader.h CSU	17
2.1.2	Parameters CSC.....	18
2.1.2.1	symbols.c CSC.....	18

	2.1.2.1.1	init_global_symbols CSU	18
	2.1.2.2	symbols.h CSU	19
2.1.3	Initialization	CSC	19
	2.1.3.1	main.c CSC	20
	2.1.3.1.1	gasp CSU	20
	2.1.3.1.2	sigh CSU	20
	2.1.3.1.3	init_stuff CSU	20
	2.1.3.1.4	read_machine_file CSU	21
	2.1.3.1.5	init_terrain_stuff CSU	21
	2.1.3.1.6	database_init CSU	21
	2.1.3.1.7	debugging_on CSU	22
	2.1.3.1.8	print_commands_from_sbx CSU	22
	2.1.3.1.9	set_terrain_dbase CSU	22
	2.1.3.1.10	set_tdb_to_cache CSU	22
	2.1.3.1.11	network_silent CSU	22
	2.1.3.1.12	set_simnet_files CSU	22
	2.1.3.1.13	set_exercise_id CSU	22
	2.1.3.1.14	isolate_connection CSU	23
	2.1.3.1.15	set_target_machine CSU	23
	2.1.3.1.16	set_number_of_fake_remotes CSU	23
	2.1.3.1.17	saf_print_help CSU	23
	2.1.3.1.18	BEGIN_ARG_TABLE(SWITCH) CSU	23
	2.1.3.1.19	main CSU	24
	2.1.3.1.20	abort_handler CSU	24
	2.1.3.1.21	exit_handler CSU	25
	2.1.3.1.22	saf_exit CSU	25
	2.1.3.2	version.c CSC	25
	2.1.3.2.1	identify_version CSU	25
2.2	SCHEDULER	CSC	26
	2.2.1	libsched CSC	27
	2.2.1.1	fncl.c CSC	27
	2.2.1.1.1	deferred_fncl CSU	27
	2.2.1.1.2	periodic_fncl CSU	28
	2.2.1.1.3	get_args CSU	28
	2.2.1.1.4	cancel_fncl CSU	28
	2.2.1.1.5	cancel_fncl_group CSU	29
	2.2.1.1.6	change_fncl_period CSU	29
	2.2.1.2	invoke.c CSC	30
	2.2.1.2.1	invoke CSU	30
	2.2.1.3	maint.c CSC	30
	2.2.1.3.1	scheduler_init CSU	31

	2.2.1.3.2	free_function CSU	31
	2.2.1.3.3	remove_function CSU	32
	2.2.1.3.4	insert_periodic_function CSU	32
	2.2.1.3.5	insert_function CSU	32
	2.2.1.3.6	which_event_ring CSU	33
2.2.1.4	perf.c CSC		33
	2.2.1.4.1	perf_monitor_on CSU	33
	2.2.1.4.2	perf_monitor_off CSU	34
	2.2.1.4.3	set_critical_performance_level CSU	34
	2.2.1.4.4	zero_perf_stats CSU	34
	2.2.1.4.5	collect_perf_stat CSU	35
	2.2.1.4.6	print_performance_stats CSU	35
2.2.1.5	sched.c CSC		35
	2.2.1.5.1	invoke_functions_until CSU	36
	2.2.1.5.2	wait_until CSU	36
2.2.1.6	libsched.h CSU		36
2.2.2	safobj.c CSC		38
	2.2.2.1	init_safobj_table CSU	38
	2.2.2.2	allocate_safobj CSU	38
	2.2.2.3	deallocate_safobj CSU	39
2.2.3	tickable.c CSC		39
	2.2.3.1	create_tickable CSU	40
	2.2.3.2	destroy_tickable CSU	40
	2.2.3.3	start_ticking CSU	40
	2.2.3.4	stop_ticking CSU	41
	2.2.3.5	change_tick_rate CSU	41
	2.2.3.6	tickable_note_start_tick CSU	41
2.2.4	saf.c CSC		42
	2.2.4.1	saf_complete_reset CSU	42
	2.2.4.2	saf_reset_for_workstation CSU	42
	2.2.4.3	saf_remove_vehicle CSU	43
	2.2.4.4	eliminate_vehicles	43
2.3	SIMNET INTERFACE CSC		44
2.3.1	pro_sim.c CSC		45
	2.3.1.1	generate_a_deactivate CSU	45
	2.3.1.2	vehicle_kill CSU	45
	2.3.1.3	vehicle_kill_remote CSU	46
	2.3.1.4	generate_indirect_fire_packet CSU	46
	2.3.1.5	vehicle_bong CSU	46
	2.3.1.6	vehicle_ping CSU	47
2.3.2	simnet.c CSC		47
	2.3.2.1	start_simnet CSU	48

2.3.2.2	simnet_tick CSU	48
2.3.2.3	AssocGetSimAddress CSU	49
2.3.2.4	ApplicationGetSiteHost CSU	49
2.3.2.5	SAF_NET_SND_KLUDGE CSU	49
2.3.2.6	host_loopback_function CSU	50
2.3.2.7	simnet_exit CSU	50
2.3.2.8	simnet_getstats CSU	51
2.3.2.9	simnet_zerostats CSU	51
2.3.2.10	enqueue_on_rcvq CSU	51
2.3.2.11	interesting_packet_test CSU	52
2.3.2.12	protocol_sim_process CSU	52
2.3.2.13	protocol_stealth_process CSU	53
2.3.2.14	protocol_data_process CSU	53
2.3.2.15	get_sender_of_last_packet CSU	53
2.4	SAF COMMAND INTERFACE CSC	54
2.4.1	SAF Command Protocol CSC	55
2.4.1.1	messages.h CSU	55
2.4.2	libudp CSC	74
2.4.2.1	udp_berkeley.c CSC	74
2.4.2.1.1	udp_open CSU	75
2.4.2.1.2	udp_close CSU	75
2.4.2.1.3	udp_show CSU	76
2.4.2.1.4	udp_read CSU	76
2.4.2.1.5	udp_write CSU	77
2.4.2.1.6	get_local_address CSU	77
2.4.2.1.7	get_remote_address CSU	77
2.4.2.1.8	print_sockaddr CSU	78
2.4.2.1.9	sockaddr_init CSU	78
2.4.2.1.10	make_me_non_blocking CSU	78
2.4.2.1.11	master_udp_open CSU	79
2.4.2.2	udp_conn.h CSU	79
2.4.2.3	rel_conn.c CSC	80
2.4.2.3.1	rudp_init CSU	81
2.4.2.3.2	rudp_show CSU	81
2.4.2.3.3	rudp_open CSU	81
2.4.2.3.4	master_rudp_open CSU	82
2.4.2.3.5	rudp_close CSU	82
2.4.2.3.6	master_rudp_close CSU	82
2.4.2.3.7	rudp_synch CSU	83
2.4.2.3.8	master_rudp_synch CSU	83
2.4.2.3.9	rudp_disconnect CSU	83
2.4.2.3.10	rudp_discard_all_buffers CSU	84
2.4.2.3.11	rudp_read_message CSU	84

	2.4.2.3.12	rudp_post_message CSU	85
	2.4.2.3.13	rudp_send CSU	85
	2.4.2.3.14	has_this_packet_been_acked CSU.....	86
	2.4.2.3.15	rudp_ack_received CSU.....	86
	2.4.2.3.16	rudp_retransmit_buf CSU.....	86
	2.4.2.3.17	rudp_retransmit CSU	87
	2.4.2.3.18	rudp_send_bare_ack CSU	87
	2.4.2.3.19	rudp_tick CSU.....	88
2.4.2.4	rel_conn.h	CSU	88
2.4.2.5	buf_conn.c	CSC.....	90
	2.4.2.5.1	buf_rudp_init CSU.....	90
	2.4.2.5.2	buf_rudp_show CSU.....	91
	2.4.2.5.3	buf_rudp_open CSU.....	91
	2.4.2.5.4	buf_master_rudp_open CSU	91
	2.4.2.5.5	buf_master_rudp_synch CSU	92
	2.4.2.5.6	buf_rudp_close CSU	92
	2.4.2.5.7	buf_master_rudp_close CSU	92
	2.4.2.5.8	buf_rudp_discard_all_buffers CSU.....	93
	2.4.2.5.9	buf_rudp_disconnect CSU.....	93
	2.4.2.5.10	buf_rudp_flush CSU	93
	2.4.2.5.11	buf_rudp_read_message CSU.....	94
	2.4.2.5.12	buf_rudp_write_message CSU.....	94
	2.4.2.5.13	buf_rudp_tick CSU	95
2.4.2.6	buf_conn.h	CSU	95
2.4.3	SAF Command Processor	CSC	96
2.4.3.1	map.c	CSC.....	96
	2.4.3.1.1	init_mappings CSU	96
	2.4.3.1.2	map_icon_symbol_to_icon_ number CSU.....	97
	2.4.3.1.3	map_echelon_symbol_to_echelon_ number CSU.....	97
	2.4.3.1.4	map_echelon_number_to_ echelon_symbol CSU.....	97
	2.4.3.1.5	map_echelon_type_symbol_to_ echelon_type_number CSU.....	98
	2.4.3.1.6	map_echelon_type_number_to_ echelon_type_symbol CSU.....	98
	2.4.3.1.7	map_echelon_type_symbol_to_ icon_number CSU	99
	2.4.3.1.8	rassoc_sym CSU.....	99
	2.4.3.1.9	icon_from_object_type CSU.....	100
2.4.3.2	sbx.c	CSC.....	101

2.4.3.2.1	open_io_connections CSU.....	103
2.4.3.2.2	exit_all_sbx_conns CSU.....	103
2.4.3.2.3	write_buffer_all_sbx CSU.....	103
2.4.3.2.4	show_connection_all_sbx CSU.....	104
2.4.3.2.5	get_sbx_from_port CSU.....	104
2.4.3.2.6	poll_request_to_string CSU.....	104
2.4.3.2.7	show_overlays_all_sbx CSU.....	105
2.4.3.2.8	sbx_printf CSU.....	105
2.4.3.2.9	sbx_connection_init CSU.....	105
2.4.3.2.10	sbx_connection_open CSU.....	106
2.4.3.2.11	sbx_connection_exit CSU.....	106
2.4.3.2.12	sbx_connection_show CSU.....	106
2.4.3.2.13	sbx_connection_synch_received CSU.....	107
2.4.3.2.14	sbx_connection_disconnect CSU.....	107
2.4.3.2.15	sbx_connection_write_buffer CSU.....	108
2.4.3.2.16	sbx_connection_overloaded CSU.....	108
2.4.3.2.17	sbx_connection_tick CSU.....	109
2.4.3.2.18	fill_sbx_opfor_header CSU.....	109
2.4.3.2.19	sbx_connection_send_to_port CSU.....	109
2.4.3.2.20	sbx_connection_process_vehicle_ impact CSU.....	110
2.4.3.2.21	sbx_connection_process_ground_ impact CSU.....	110
2.4.3.2.22	sbx_connection_process_indirect_ fire CSU.....	111
2.4.3.2.23	indirect_fire_to_sbx CSU.....	111
2.4.3.2.24	ground_impact_to_sbx CSU.....	112
2.4.3.2.25	vehicle_impact_to_sbx CSU.....	112
2.4.3.2.26	sbx_connection_process_message CSU.....	112
2.4.3.2.27	sbx_connection_process_ messages CSU.....	114
2.4.3.2.28	country_to_string CSU.....	114
2.4.3.2.29	sbx_connection_create_unit_msg CSU.....	115
2.4.3.2.30	sbx_connection_reset_msg CSU.....	115
2.4.3.2.31	sbx_connection_generate_reset_ vehicle CSU.....	115
2.4.3.2.32	sbx_connection_vehicle_reinit CSU.....	116

2.4.3.2.33	sbx_connection_service_poll_msg CSU.....	116
2.4.3.2.34	sbx_connection_wat_pulse CSU.....	117
2.4.3.2.35	sbx_connection_stat_pulse CSU.....	118
2.4.3.2.36	arty_type_string CSU.....	118
2.4.3.2.37	sbx_connection_generate_arty_ msg CSU.....	119
2.4.3.2.38	sbx_connection_query_sub_state CSU.....	119
2.4.3.2.39	sbx_connection_set_ivis_xmit_ modes CSU	120
2.4.3.2.40	sbx_connection_set_ivis_ parameters CSU	120
2.4.3.2.41	sbx_swap_known_vehicles CSU	120
2.4.3.2.42	sbx_set_all_known_vehicles CSU....	121
2.4.3.2.43	sbx_set_specific_known_vehicles CSU.....	121
2.4.3.2.44	sbx_set_top_level_known_ vehicles CSU	121
2.4.3.2.45	sbx_add_point CSU	122
2.4.3.2.46	sbx_add_area CSU.....	122
2.4.3.2.47	sbx_add_zone CSU.....	123
2.4.3.2.48	sbx_add_line CSU	123
2.4.3.2.49	sbx_add_route CSU	123
2.4.3.2.50	sbx_delete_overlay CSU	124
2.4.3.2.51	sbx_execute_overlay CSU	124
2.4.3.2.52	sbx_halt CSU.....	124
2.4.3.2.53	sbx_change_speed CSU.....	125
2.4.3.2.54	sbx_change_formation CSU.....	125
2.4.3.2.55	sbx_delete_cm CSU	126
2.4.3.2.56	sbx_rejoin_unit CSU	126
2.4.3.2.57	sbx_follow_vehicle CSU.....	126
2.4.3.2.58	sbx_simulator_in_command CSU	127
2.4.3.2.59	sbx_goto_point CSU	127
2.4.3.2.60	sbx_resume_mission CSU.....	128
2.4.3.2.61	sbx_face_direction CSU.....	128
2.4.3.2.62	sbx_set_targeting_parameters CSU.....	129
2.4.3.2.63	sbx_land CSU	129
2.4.3.2.64	string_for_altitude_type CSU.....	129
2.4.3.2.65	sbx_altitude CSU.....	130
2.4.3.2.66	string_for_hold_type CSU.....	130
2.4.3.2.67	sbx_hold CSU.....	130
2.4.3.2.68	string_for_attack_type CSU	131

	2.4.3.2.69	sbx_attack CSU	131
	2.4.3.2.70	sbx_attach_stealth CSU	132
	2.4.3.2.71	sbx_resupply CSU.....	132
	2.4.3.2.72	broadcast_echelon_data CSU.....	133
	2.4.3.2.73	broadcast_appearance_data CSU.....	133
	2.4.3.2.74	broadcast_pae_data CSU.....	134
	2.4.3.2.75	broadcast_veh_is_gone CSU	134
	2.4.3.2.76	print_opfor_header CSU	135
	2.4.3.2.77	print_message_position CSU.....	135
	2.4.3.2.78	print_vehicle_ids CSU.....	135
	2.4.3.2.79	sbx_connection_show_top_level_	
		units CSU	135
	2.4.3.3	sbx.h CSU	136
2.5	PARSER INTERFACE CSC.....		137
2.5.1	libparser CSC.....		138
	2.5.1.1	par_base.c CSC	138
		2.5.1.1.1 ParseInput CSU	138
		2.5.1.1.2 ParseList CSU.....	139
		2.5.1.1.3 ParseKeyword CSU	140
		2.5.1.1.4 ParseCommandDone CSU.....	141
		2.5.1.1.5 ParseDoKeywordTable CSU	142
	2.5.1.2	par_const.c CSC	142
		2.5.1.2.1 ParseGetConstant CSU.....	142
	2.5.1.3	par_copy.c CSC.....	143
		2.5.1.3.1 stringcopy CSU	143
	2.5.1.4	par_edit.c CSC	144
		2.5.1.4.1 ChangePrompt CSU	145
		2.5.1.4.2 InitParser CSU	145
		2.5.1.4.3 UpdateLine CSU	145
		2.5.1.4.4 UpdateMoveCursor CSU	146
		2.5.1.4.5 RedisplayLine CSU.....	146
		2.5.1.4.6 SetStopPoint CSU	146
		2.5.1.4.7 NextChar CSU	147
		2.5.1.4.8 MarkCursor CSU	147
		2.5.1.4.9 RestoreCursor CSU.....	147
		2.5.1.4.10 EmptyLine CSU.....	147
		2.5.1.4.11 BeginningOfLine CSU	148
		2.5.1.4.12 Back1Char CSU.....	148
		2.5.1.4.13 Delete1Forward CSU.....	148
		2.5.1.4.14 EndOfLine CSU.....	149
		2.5.1.4.15 Forward1Char CSU	149
		2.5.1.4.16 GobbleWord CSU	149
		2.5.1.4.17 Delete1Backward CSU.....	149

	2.5.1.4.18	KillForward CSU	150
	2.5.1.4.19	ClearLine CSU	150
	2.5.1.4.20	LeftWord CSU	150
	2.5.1.4.21	RightWord CSU.....	151
	2.5.1.4.22	DeleteWord CSU.....	151
	2.5.1.4.23	InsertChar CSU	151
	2.5.1.4.24	ForwardChar CSU.....	152
	2.5.1.4.25	BackChar CSU	152
	2.5.1.4.26	DeleteChar CSU.....	152
	2.5.1.4.27	ScanForward CSU.....	153
	2.5.1.4.28	LookForward CSU	153
	2.5.1.4.29	ScanBackwards CSU.....	153
2.5.1.5	par_fields.c	CSC	154
	2.5.1.5.1	ParseGetFields CSU.....	154
	2.5.1.5.2	TypeFields CSU.....	155
	2.5.1.5.3	FieldGet CSU.....	155
2.5.1.6	par_lib.c	CSC.....	156
	2.5.1.6.1	ParseGetDecimal CSU.....	156
	2.5.1.6.2	ParseGetHex CSU	157
	2.5.1.6.3	ParseGetOctal CSU	157
	2.5.1.6.4	ParseGetString CSU.....	158
	2.5.1.6.5	ParsePutArg CSU.....	158
	2.5.1.6.6	ParseOptional CSU	159
	2.5.1.6.7	ParseConvertNumber CSU	159
2.5.1.7	par_util.c	CSC.....	160
	2.5.1.7.1	DoCommand CSU	160
	2.5.1.7.2	DoHelp CSU.....	161
	2.5.1.7.3	DoEscape CSU	161
	2.5.1.7.4	ParseMustFree CSU	161
	2.5.1.7.5	ParseInSet CSU	162
	2.5.1.7.6	ParseGetToken CSU.....	162
	2.5.1.7.7	ParseError CSU	163
	2.5.1.7.8	ParseMessage CSU	163
	2.5.1.7.9	ParseFindEndList CSU.....	163
	2.5.1.7.10	ParseMatch CSU	164
	2.5.1.7.11	ParseTableFind CSU	164
	2.5.1.7.12	ParseEscapeComplete CSU.....	165
	2.5.1.7.13	uc CSU	165
	2.5.1.8	par_hist.c CSC	166
	2.5.1.8.1	CommandLog CSU.....	166
	2.5.1.8.2	CopyEarlierCommand CSU	166
	2.5.1.8.3	PreviousCommand CSU	167
	2.5.1.8.4	NextCommand CSU.....	167

2.5.1.9	par_unix.c CSC	168
2.5.1.9.1	Alloc CSU.....	168
2.5.1.9.2	Free CSU.....	168
2.5.1.9.3	ParsePrint CSU.....	169
2.5.1.10	kludge.c CSU.....	169
2.5.1.11	libparser.h CSU	169
2.5.2	Parser Command Processor CSC.....	171
2.5.2.1	parser.c CSC.....	171
2.5.2.1.1	parser_init CSU	173
2.5.2.1.2	parser_restore_term CSU	173
2.5.2.1.3	parser_create CSU	173
2.5.2.1.4	set_command_printing CSU.....	174
2.5.2.1.5	set_abort_on_error CSU	174
2.5.2.1.6	set_ground_impact_mode CSU	174
2.5.2.1.7	set_indirect_fire_mode CSU.....	175
2.5.2.1.8	set_header_printing CSU.....	175
2.5.2.1.9	set_monitor_period CSU.....	175
2.5.2.1.10	toggle_debugging CSU.....	176
2.5.2.1.11	show_vehicle_ids CSU.....	176
2.5.2.1.12	count_vehicles CSU	176
2.5.2.1.13	count_forces CSU.....	177
2.5.2.1.14	count_sites CSU.....	177
2.5.2.1.15	count_hosts CSU.....	177
2.5.2.1.16	show_connection CSU	178
2.5.2.1.17	show_sbx_overlays CSU	178
2.5.2.1.18	identify_exercise CSU.....	179
2.5.2.1.19	lookup_vehicle_with_range_check CSU.....	179
2.5.2.1.20	show_vehicle CSU.....	179
2.5.2.1.21	vehicle_catastrophe CSU.....	180
2.5.2.1.22	vehicle_resupply CSU.....	180
2.5.2.1.23	vehicle_fake_resupply CSU	180
2.5.2.1.24	vehicle_defuel CSU.....	181
2.5.2.1.25	vehicle_ping_do CSU	181
2.5.2.1.26	vehicle_bong_do CSU.....	181
2.5.2.1.27	stealth_set_symbols_draw_tick CSU.....	182
2.5.2.1.28	stealth_set_mimic_on CSU	182
2.5.2.1.29	stealth_set_mimic_off CSU.....	182
2.5.2.1.30	stealth_site_host_pair CSU	182
2.5.2.1.31	stealth_attach_to CSU	182
2.5.2.1.32	stealth_teleport_to CSU	183
2.5.2.1.33	parser_create_vehicle CSU.....	183

	2.5.2.1.34	countries_from_battle_scheme_	
		and_force CSU	184
	2.5.2.1.35	parser_global_reset CSU	184
	2.5.2.1.36	parser_heap_statistics CSU	184
	2.5.2.1.37	parser_heap_verify CSU	185
	2.5.2.1.38	print_reasons_and_clear CSU	185
	2.5.2.1.39	parser_heap_collect CSU	186
	2.5.2.1.40	parser_set_targeting_parameters	
		CSU	186
	2.5.2.1.41	parser_send_string CSU	186
	2.5.2.2	debug.h CSU	195
2.6	LOCAL VEHICLES CSC		197
2.6.1	Local Vehicles Main CSC		198
	2.6.1.1	saf_vehicle.c CSC	199
	2.6.1.1.1	create_saf_vehicle CSU	200
	2.6.1.1.2	saf_vehicle_go_away CSU	201
	2.6.1.1.3	saf_vehicle_start_ticking CSU	202
	2.6.1.1.4	saf_vehicle_set_superior CSU	202
	2.6.1.1.5	saf_vehicle_set_leader CSU	202
	2.6.1.1.6	saf_vehicle_teleport_to_station	
		CSU	203
	2.6.1.1.7	saf_vehicle_teleport CSU	203
	2.6.1.1.8	get_guises CSU	204
	2.6.1.1.9	saf_vehicle_set_marking CSU	204
	2.6.1.1.10	vehicle_status_string CSU	204
	2.6.1.1.11	saf_vehicle_show CSU	205
	2.6.1.1.12	saf_vehicle_set_targeting_	
		parameters CSU	205
	2.6.1.1.13	saf_vehicle_fill_in_appearance_	
		data CSU	206
	2.6.1.1.14	saf_vehicle_fill_in_echelon_data	
		CSU	206
	2.6.1.1.15	saf_vehicle_fill_in_position_data	
		CSU	206
	2.6.1.1.16	saf_vehicle_checkpoint_state	
		CSU	207
	2.6.1.1.17	saf_vehicle_reinit CSU	207
	2.6.1.1.18	saf_vehicle_mimic_vehicle CSU	208
	2.6.1.1.19	saf_vehicle_stop_mimicing CSU	208
	2.6.1.1.20	saf_vehicle_tick CSU	208
	2.6.1.1.21	saf_vehicle_send_appearance	
		CSU	209
	2.6.1.1.22	saf_vehicle_firepower_kill CSU	210
	2.6.1.1.23	saf_vehicle_mobility_kill CSU	210

2.6.1.1.24	saf_vehicle_stop_flaming CSU.....	210
2.6.1.1.25	saf_vehicle_catastrophic_kill CSU....	211
2.6.1.1.26	saf_vehicle_drain_supplies CSU.....	211
2.6.1.1.27	saf_vehicle_sudden_stop CSU.....	211
2.6.1.1.28	saf_vehicle_next_event_id CSU.....	211
2.6.1.1.29	saf_vehicle_supplies_needed CSU.....	212
2.6.1.1.30	saf_vehicle_supplies_provided CSU.....	212
2.6.1.1.31	saf_vehicle_out_of_gas CSU.....	212
2.6.1.1.32	saf_vehicle_out_of_ammo CSU.....	213
2.6.1.1.33	saf_vehicle_fake_resupply CSU.....	213
2.6.1.1.34	saf_vehicle_generate_status_report CSU.....	213
2.6.1.1.35	saf_vehicle_remove_vehicles CSU.....	214
2.6.1.1.36	saf_vehicle_vehicle_impact CSU.....	215
2.6.1.1.37	saf_vehicle_vehicle_rammed CSU....	215
2.6.1.1.38	saf_vehicle_indirect_fire CSU.....	215
2.6.1.1.39	saf_vehicle_pro_sim CSU.....	216
2.6.1.1.40	saf_vehicle_mission_completed CSU.....	216
2.6.1.1.41	saf_vehicle_useless CSU.....	217
2.6.1.1.42	saf_vehicle_collision_over CSU.....	217
2.6.1.1.43	saf_vehicle_doing_collision_stuff CSU.....	217
2.6.1.1.44	saf_vehicle_simulator_in_ command CSU.....	218
2.6.1.1.45	saf_vehicle_rejoin_unit CSU.....	218
2.6.1.1.46	saf_vehicle_execute_overlay CSU....	219
2.6.1.1.47	saf_vehicle_cancel_overlay CSU.....	219
2.6.1.1.48	saf_vehicle_set_route CSU.....	219
2.6.1.1.49	saf_vehicle_set_speed CSU.....	220
2.6.1.1.50	saf_vehicle_set_direction CSU.....	220
2.6.1.1.51	saf_vehicle_reset_station_keeper CSU.....	221
2.6.1.1.52	saf_vehicle_halt CSU.....	221
2.6.1.1.53	saf_vehicle_change_speed CSU.....	221
2.6.1.1.54	saf_vehicle_follow_vehicle CSU.....	222
2.6.1.1.55	saf_vehicle_goto_point CSU.....	222
2.6.1.1.56	saf_vehicle_resume_mission CSU....	223
2.6.1.1.57	saf_vehicle_face_direction CSU.....	223
2.6.1.1.58	saf_vehicle_est_position CSU.....	223
2.6.2	Damage CSC	224

2.6.2.1	damage.c CSC.....	224
2.6.2.1.1	create_damage CSU.....	225
2.6.2.1.2	destroy_damage CSU.....	225
2.6.2.1.3	damage_string CSU.....	225
2.6.2.1.4	damage_vehicle_impact CSU.....	226
2.6.2.1.5	damage_vehicle_rammed CSU	226
2.6.2.1.6	damage_indirect_fire CSU	227
2.6.3	Ground Maneuver CSC	228
2.6.3.1	collision.c CSC.....	228
2.6.3.1.1	create_collision CSU.....	229
2.6.3.1.2	destroy_collision CSU.....	229
2.6.3.1.3	collision_remove_vehicles CSU.....	230
2.6.3.1.4	collision_show CSU.....	230
2.6.3.1.5	collision_disengaging CSU	230
2.6.3.1.6	collision_tick CSU	231
2.6.3.1.7	collision_dead_tick CSU	232
2.6.3.1.8	detect_imminent_collision_tick CSU.....	232
2.6.3.1.9	start_avoiding_collision CSU.....	233
2.6.3.1.10	detect_building_on_path_tick CSU.....	233
2.6.3.1.11	start_avoiding_house CSU.....	234
2.6.3.1.12	collision_vehicle_rammed CSU	234
2.6.3.1.13	detect_collision_tick CSU.....	234
2.6.3.1.14	start_disengaging CSU	235
2.6.3.1.15	phase_two_collision_check CSU	236
2.6.3.1.16	choose_skirt_point CSU	236
2.6.3.2	driver.c CSC.....	237
2.6.3.2.1	create_driver CSU.....	238
2.6.3.2.2	destroy_driver CSU.....	239
2.6.3.2.3	driver_remove_vehicles CSU.....	239
2.6.3.2.4	driver_tick CSU	239
2.6.3.2.5	stop CSU.....	240
2.6.3.2.6	set_speed_dir CSU.....	240
2.6.3.2.7	next_route_point CSU.....	241
2.6.3.2.8	vel2point CSU.....	241
2.6.3.2.9	mission_stationpoint CSU	241
2.6.3.2.10	stationpoint CSU	242
2.6.3.2.11	get_unit_direction CSU.....	242
2.6.3.2.12	driver_simulator_in_command CSU.....	243
2.6.3.2.13	get_leader_state CSU	243
2.6.3.2.14	followvehicle CSU.....	244

	2.6.3.2.15	driver_executing_immediate_	
		command CSU	244
	2.6.3.2.16	driver_mission_completed CSU.....	245
	2.6.3.2.17	driver_execute_overlay CSU	245
	2.6.3.2.18	driver_set_leader_mis CSU.....	246
	2.6.3.2.19	driver_follow_leader CSU	246
	2.6.3.2.20	driver_stop_mission CSU.....	246
	2.6.3.2.21	driver_forget_about_forps CSU.....	247
	2.6.3.2.22	driver_set_route CSU.....	247
	2.6.3.2.23	driver_set_speed CSU.....	247
	2.6.3.2.24	driver_set_direction CSU	247
	2.6.3.2.25	driver_set_routedirection CSU	248
	2.6.3.2.26	driver_resume_from_collision	
		CSU.....	248
	2.6.3.2.27	driver_halt_cmd CSU.....	248
	2.6.3.2.28	driver_follow_vehicle_cmd CSU	248
	2.6.3.2.29	driver_goto_point_cmd CSU	249
	2.6.3.2.30	driver_face_direction_cmd CSU	249
	2.6.3.2.31	driver_resume_mission_cmd	
		CSU.....	250
	2.6.3.2.32	driver_resume_cmd CSU	250
	2.6.3.2.33	driver_change_speed_cmd CSU	250
	2.6.3.2.34	driver_set_direction_cmd CSU.....	250
	2.6.3.2.35	driver_set_route_direction_cmd	
		CSU.....	251
	2.6.3.2.36	vec_set CSU	251
	2.6.3.2.37	vec2_veh2world CSU	251
	2.6.3.2.38	vec2_world2veh CSU	252
	2.6.3.2.39	printmission CSU	252
	2.6.3.2.40	printdirection CSU	252
	2.6.3.2.41	printroutedirection CSU	252
	2.6.3.2.42	printimmediate CSU	253
	2.6.3.2.43	driver_show CSU.....	253
2.6.3.3		driver.h CSU	254
2.6.3.4		groundveh.c CSC.....	255
	2.6.3.4.1	create_groundveh CSU.....	255
	2.6.3.4.2	destroy_groundveh CSU.....	256
	2.6.3.4.3	groundveh_move CSU	256
	2.6.3.4.4	groundveh_tick CSU	257
	2.6.3.4.5	groundveh_mobility_kill CSU.....	257
	2.6.3.4.6	groundveh_close_enough CSU ...	258
2.6.4		Air Maneuver CSC.....	259
	2.6.4.1	helo.c CSC.....	259
	2.6.4.1.1	helo_tick CSU	260

2.6.4.2	pilot.c CSC.....	261
2.6.4.2.1	pilot_state_to_string CSU.....	264
2.6.4.2.2	create_pilot CSU	264
2.6.4.2.3	init_pilot_state_machine CSU	265
2.6.4.2.4	destroy_pilot CSU	265
2.6.4.2.5	pilot_remove_vehicles CSU	265
2.6.4.2.6	pilot_show_machine CSU	266
2.6.4.2.7	pilot_show CSU.....	266
2.6.4.2.8	ground_level CSU	266
2.6.4.2.9	agl_to_abs_altitude CSU	267
2.6.4.2.10	dir_to_tan_point CSU	267
2.6.4.2.11	waypoint_vel CSU.....	268
2.6.4.2.12	orbit_velocity CSU.....	269
2.6.4.2.13	z_velocity CSU.....	269
2.6.4.2.14	combined_velocity CSU.....	270
2.6.4.2.15	pilot_on_same_route CSU	270
2.6.4.2.16	pilot_is_facing_direction CSU.....	270
2.6.4.2.17	vehicle_facing_point CSU	271
2.6.4.2.18	are_we_there CSU	271
2.6.4.2.19	xy_dir_and_range CSU	272
2.6.4.2.20	compute_situation CSU	272
2.6.4.2.21	project_point CSU	273
2.6.4.2.22	vec2_rotate CSU	273
2.6.4.2.23	get_next_circle_point CSU.....	274
2.6.4.2.24	pilot_start_takingoff CSU.....	274
2.6.4.2.25	pilot_takingoff CSU	275
2.6.4.2.26	pilot_start_gotopoint CSU	275
2.6.4.2.27	pilot_gotopoint CSU.....	275
2.6.4.2.28	pilot_goto_endpoint CSU	276
2.6.4.2.29	pilot_hold CSU.....	277
2.6.4.2.30	pilot_init_hoverhold CSU.....	277
2.6.4.2.31	pilot_start_hoverhold CSU.....	277
2.6.4.2.32	pilot_hoverhold CSU	278
2.6.4.2.33	pilot_hoverhold_tick CSU	278
2.6.4.2.34	pilot_init_orbithold CSU	278
2.6.4.2.35	pilot_start_orbithold CSU.....	279
2.6.4.2.36	pilot_orbithold CSU	279
2.6.4.2.37	pilot_orbit_tick CSU.....	279
2.6.4.2.38	pilot_racetrackhold CSU	280
2.6.4.2.39	pilot_init_land CSU.....	280
2.6.4.2.40	pilot_start_landing CSU.....	281
2.6.4.2.41	pilot_landing CSU	281
2.6.4.2.42	pilot_start_landed CSU.....	281

2.6.4.2.43	pilot_landhold_tick CSU	282
2.6.4.2.44	pilot_init_followroute CSU	282
2.6.4.2.45	pilot_followroute_tick CSU.....	283
2.6.4.2.46	pilot_init_hoverattack CSU	283
2.6.4.2.47	pilot_start_hoverattack_approach CSU.....	284
2.6.4.2.48	pilot_hoverattack_approach CSU	284
2.6.4.2.49	pilot_start_hoverattack CSU	285
2.6.4.2.50	pilot_point_at_target CSU.....	285
2.6.4.2.51	pilot_hoverattack CSU.....	285
2.6.4.2.52	pilot_start_hoverattack_egress CSU.....	286
2.6.4.2.53	pilot_start_hoverattack_complete CSU.....	286
2.6.4.2.54	pilot_hoverattack_complete CSU.....	287
2.6.4.2.55	pilot_hoverattack_tick CSU.....	287
2.6.4.2.56	pilot_follow_vehicle CSU.....	288
2.6.4.2.57	pilot_followvehicle CSU	288
2.6.4.2.58	pilot_face_direction CSU.....	288
2.6.4.2.59	pilot_tick2 CSU	289
2.6.4.2.60	idle_tick CSU.....	289
2.6.4.2.61	attackatwill_tick CSU.....	290
2.6.4.2.62	attackatwill_tick_new CSU	290
2.6.4.2.63	pilot_check_state CSU.....	291
2.6.4.2.64	pilot_tick CSU.....	291
2.6.4.2.65	pilot_get_altitude CSU.....	291
2.6.4.2.66	pilot_change_altitude_im CSU	292
2.6.4.2.67	pilot_get_speed CSU	292
2.6.4.2.68	pilot_change_speed_im CSU	292
2.6.4.2.69	pilot_goto_point_im CSU.....	293
2.6.4.2.70	pilot_land_im CSU.....	293
2.6.4.2.71	pilot_cancel_immediate CSU	293
2.6.4.2.72	pilot_executing_immediate_ command CSU	294
2.6.4.2.73	pilot_follow_vehicle_im CSU	294
2.6.4.2.74	pilot_face_direction_im CSU	294
2.6.4.2.75	pilot_hoverattack_im CSU.....	295
2.6.4.2.76	pilot_hold_im CSU	295
2.6.4.2.77	pilot_mission_completed CSU	295
2.6.4.2.78	pilot_stationpoint CSU	296
2.6.4.2.79	pilot_set_leader_mis CSU.....	296
2.6.4.2.80	pilot_follow_leader CSU... ..	296
2.6.4.2.81	pilot_set_route_mis CSU	297
2.6.4.2.82	pilot_stop_mission CSU	297

	2.6.4.2.83	pilot_set_speed_mis CSU.....	297
	2.6.4.2.84	pilot_get_asm CSU	298
	2.6.4.2.85	pilot_execute_overlay CSU	298
2.6.4.3	p_follower.c	CSC.....	299
	2.6.4.3.1	p_follower_am_i_flying_coord CSU.....	299
	2.6.4.3.2	p_follow_form_flip CSU	300
	2.6.4.3.3	p_follower_fly_in_coord_pos CSU.....	300
	2.6.4.3.4	p_follower_comp_reset_route CSU.....	301
	2.6.4.3.5	p_follower_arrive_at_same_time CSU.....	301
	2.6.4.3.6	p_follower_set_desired_vel CSU	301
	2.6.4.3.7	p_follower_get_wrld_offset CSU.....	302
	2.6.4.3.8	p_follower_stop_coord CSU	303
	2.6.4.3.9	p_follower_gen_coord_goal_pnt CSU.....	303
	2.6.4.3.10	p_follower_set_follow CSU.....	304
	2.6.4.3.11	p_follower_new_rel_rt_pnt CSU	304
	2.6.4.3.12	p_follower_flying_independent CSU.....	304
	2.6.4.3.13	p_follower_followroute_tick CSU....	305
	2.6.4.3.14	p_follower_leader_passed_point CSU.....	306
	2.6.4.3.15	p_follower_passed_point CSU.....	306
	2.6.4.3.16	p_follower_hoverattack_tick CSU....	307
	2.6.4.3.17	p_follower_goto_pnt_coord CSU.....	307
	2.6.4.3.18	p_follower_hoverhold CSU	308
	2.6.4.3.19	p_follower_land CSU	308
	2.6.4.3.20	p_follower_landhold_tick CSU	309
	2.6.4.3.21	p_follower_hoverhold_tick CSU.....	309
	2.6.4.3.22	pilot_follow_leader_tick CSU	310
	2.6.4.3.23	p_follower_flip_in_turn CSU	311
	2.6.4.3.24	p_follower_find_pnt_in_turn CSU.....	311
	2.6.4.3.25	p_follower_pnt_in_no_turn CSU	312
	2.6.4.3.26	p_follower_pnt_in_shallow_turn CSU.....	312
	2.6.4.3.27	p_follower_pnt_in_hard_turn CSU.....	313
2.6.4.4	plane.c	CSC.....	313
	2.6.4.4.1	plane_tick CSU.....	314
2.6.4.5	impact.c	CSC.....	315
	2.6.4.5.1	get_impact_and_trajectory CSU.....	316

	2.6.4.5.2	get_i_and_t_from_normal CSU	316
	2.6.4.5.3	randomize_vector CSU	317
2.6.4.6	flyingveh.c	CSC	317
	2.6.4.6.1	airveh_init CSU	318
	2.6.4.6.2	create_airveh CSU	318
	2.6.4.6.3	destroy_airveh CSU	319
	2.6.4.6.4	airveh_show CSU	319
	2.6.4.6.5	air_tick CSU	319
	2.6.4.6.6	airveh_mobility_kill CSU	320
	2.6.4.6.7	airveh_catastrophic_kill CSU	320
	2.6.4.6.8	hull_to_world_from_orientation CSU	320
	2.6.4.6.9	init_static_matrices CSU	320
	2.6.4.6.10	induce_tail_spin CSU	321
	2.6.4.6.11	induce_roll CSU	321
2.6.4.7	flyingveh.h	CSU	321
2.6.5	Intervisibility	CSC	326
2.6.5.1	detection.c	CSC	326
	2.6.5.1.1	create_detection CSU	327
	2.6.5.1.2	clear_detection CSU	327
	2.6.5.1.3	destroy_detection CSU	327
	2.6.5.1.4	detection_remove_vehicles CSU	327
	2.6.5.1.5	detection_show CSU	328
	2.6.5.1.6	detection_show_type CSU	328
	2.6.5.1.7	compute_enemy_weight CSU	328
	2.6.5.1.8	compute_interest_direction CSU	329
	2.6.5.1.9	arc_of_attention CSU	329
	2.6.5.1.10	detectable CSU	329
	2.6.5.1.11	detection_tick CSU	330
	2.6.5.1.12	detection_save_weight CSU	330
	2.6.5.1.13	major_detection_increase CSU	331
2.6.5.2	intervis.c	CSC	331
	2.6.5.2.1	intervis_can_see_pt_to_pt CSU	332
	2.6.5.2.2	intervis_get_view CSU	332
	2.6.5.2.3	intervis_possibly_visible CSU	333
	2.6.5.2.4	intervis_pt_to_pt CSU	333
	2.6.5.2.5	intervis_get_high_ground CSU	334
	2.6.5.2.6	set_pv_params CSU	334
2.6.6	Resupply	CSC	334
2.6.6.1	logistics.c	CSC	335
	2.6.6.1.1	create_logistics CSU	335
	2.6.6.1.2	destroy_logistics CSU	336
	2.6.6.1.3	logistics_remove_vehicles CSU	336

	2.6.6.1.4	start_resupply_of_to CSU	336
	2.6.6.1.5	resupply_check_ok CSU	337
	2.6.6.1.6	resupply_bld_ammo_needs CSU	337
	2.6.6.1.7	choose_resupply_item CSU	338
	2.6.6.1.8	logistics_supply_offer_canceled CSU	338
	2.6.6.1.9	logistics_supply_offer_received CSU	338
	2.6.6.1.10	logistics_tick CSU	339
	2.6.6.2	logistics.h CSU	340
2.6.7	libpvis CSC		340
	2.6.7.1	pve_checkvis.c CSC	341
	2.6.7.1.1	pve_checkvis CSU	344
	2.6.7.1.2	startup CSU	345
	2.6.7.1.3	clip_to_tdb CSU	345
	2.6.7.1.4	test_mins CSU	346
	2.6.7.1.5	test_maxima CSU	346
	2.6.7.1.6	patch CSU	347
	2.6.7.1.7	terrain CSU	347
	2.6.7.1.8	check_edges CSU	348
	2.6.7.1.9	get_mid_pt CSU	348
	2.6.7.1.10	compute_mid CSU	349
	2.6.7.1.11	check_edge_hit CSU	349
	2.6.7.1.12	objects CSU	350
	2.6.7.1.13	obstacle CSU	350
	2.6.7.1.14	check_object CSU	350
	2.6.7.1.15	trees CSU	351
	2.6.7.1.16	treelines CSU	351
	2.6.7.1.17	report_tree_block CSU	352
	2.6.7.1.18	canopies CSU	352
	2.6.7.1.19	store_hit CSU	352
	2.6.7.1.20	average CSU	353
	2.6.7.1.21	check_hits CSU	353
	2.6.7.1.22	report_hit CSU	354
	2.6.7.1.23	print_ch_pt_info CSU	356
	2.6.7.1.24	add_last_hit CSU	356
	2.6.7.1.25	report_last_hit CSU	356
	2.6.7.1.26	test_clutter CSU	357
	2.6.7.1.27	grid_locword CSU	357
	2.6.7.1.28	fix_coords CSU	358
	2.6.7.1.29	check_tree_hit CSU	358
	2.6.7.1.30	diffraction_diff CSU	358
	2.6.7.1.31	check_box CSU	359

	2.6.7.1.32	count_vtx1_hit CSU	359
	2.6.7.1.33	count_vtx2_hit CSU	360
	2.6.7.1.34	count_mid_hit CSU	360
	2.6.7.1.35	report_edge_hit CSU	360
	2.6.7.1.36	print_edge CSU	360
	2.6.7.1.37	edge_glw_miss CSU	361
	2.6.7.1.38	edge_z_miss CSU	361
	2.6.7.1.39	sort_trim_hits CSU	361
	2.6.7.1.40	print_object_info CSU	362
	2.6.7.1.41	debug_report CSU	362
	2.6.7.1.42	print_hitpt CSU	362
	2.6.7.1.43	print_fpe_info CSU	362
	2.6.7.1.44	print_pvparams CSU	363
	2.6.7.1.45	print_short_pvparams CSU	363
	2.6.7.1.46	print_error CSU	363
	2.6.7.1.47	print_debug_list CSU	364
	2.6.7.1.48	attenuation_factor CSU	364
	2.6.7.1.49	print_guard_info CSU	364
	2.6.7.1.50	print_patch_info CSU	365
	2.6.7.1.51	skyline_error CSU	365
	2.6.7.1.52	print_end_info CSU	365
	2.6.7.1.53	vis_code CSU	366
	2.6.7.1.54	report_vis_change CSU	366
	2.6.7.1.55	get_object_name CSU	366
	2.6.7.1.56	interpolate CSU	367
	2.6.7.1.57	print_terrain_point CSU	368
	2.6.7.2	pv_bv.c CSC	368
	2.6.7.2.1	point_in_bv CSU	368
	2.6.7.2.2	in_object2 CSU	369
	2.6.7.3	pve.h CSU	370
	2.6.7.4	pvis_call.h CSU	373
2.6.8	libdatabase	CSC	376
	2.6.8.1	interpolate.c CSC	376
	2.6.8.1.1	interpolate_curve CSU	376
	2.6.8.2	detection.c CSC	377
	2.6.8.2.1	show_detection_database CSU	377
	2.6.8.2.2	database_detection_query CSU	378
	2.6.8.3	hitmodel.c CSC	378
	2.6.8.3.1	show_hitmodel_database CSU	378
	2.6.8.4	formation.c CSC	379
	2.6.8.4.1	database_read_formation CSU	379
	2.6.8.4.2	database_formation_form_query CSU	379

	2.6.8.4.3	database_formation_sub_form_	
		query CSU	380
2.6.8.5	echelon.c	CSC.....	380
	2.6.8.5.1	database_read_echelon CSU	381
	2.6.8.5.2	database_echelon_template_query	
		CSU	381
	2.6.8.5.3	database_echelon_lobj_query	
		CSU	381
	2.6.8.5.4	database_echelon_response_	
		formation_query CSU	382
2.6.8.6	df_damage.c	CSC.....	383
	2.6.8.6.1	show_df_damage_database CSU	383
	2.6.8.6.2	show_df_damage_veh CSU	384
	2.6.8.6.3	show_df_damage_weapon CSU	384
	2.6.8.6.4	show_df_damage_object CSU	384
	2.6.8.6.5	show_df_damage_side CSU	384
	2.6.8.6.6	show_df_damage_entry CSU	385
	2.6.8.6.7	vehicle_wall_name CSU	385
	2.6.8.6.8	vehicle_component_name CSU	385
	2.6.8.6.9	sort_damage_table CSU	386
	2.6.8.6.10	database_df_damage_query CSU	386
	2.6.8.6.11	compute_damage_keys CSU	387
2.6.8.7	if_damage.c	CSC.....	387
	2.6.8.7.1	show_if_damage_database CSU	387
	2.6.8.7.2	show_if_damage_veh CSU	388
	2.6.8.7.3	show_if_damage_weapon CSU	388
	2.6.8.7.4	show_if_ranged_damage_entry	
		CSU	388
	2.6.8.7.5	show_if_damage_entry CSU	388
	2.6.8.7.6	database_if_damage_query CSU	389
2.6.8.8	database.c	CSC.....	389
	2.6.8.8.1	database_read CSU	390
	2.6.8.8.2	show_table_record CSU	390
	2.6.8.8.3	show_curves CSU	390
2.6.8.9	libdatabase.h	CSU	390
2.6.9	Weapons	CSC.....	392
	2.6.9.1	missile.c	CSC.....
			392
	2.6.9.1.1	hull_to_world_from_direction	
		CSU	393
	2.6.9.1.2	missile_state_to_string CSU	393
	2.6.9.1.3	create_missile CSU	393
	2.6.9.1.4	destroy_missile CSU	394
	2.6.9.1.5	missile_show CSU	394
	2.6.9.1.6	fire_missile_at_target CSU	395

	2.6.9.1.7	missile_ground_impact CSU	395
	2.6.9.1.8	missile_deactivate CSU.....	396
	2.6.9.1.9	compute_explosion_point CSU	396
	2.6.9.1.10	missile_maybe_hit_target CSU.....	397
	2.6.9.1.11	fly_missile CSU.....	397
	2.6.9.1.12	missile_set_desired_direction CSU.....	398
	2.6.9.1.13	missile_send_appearance CSU	399
2.6.9.2		missile.h CSU	399
2.6.9.3		targeting.c CSC.....	400
	2.6.9.3.1	map_role_sym_to_role_number CSU.....	401
	2.6.9.3.2	create_targeting CSU	402
	2.6.9.3.3	destroy_targeting CSU	402
	2.6.9.3.4	firestatus_to_string CSU	402
	2.6.9.3.5	targeting_show CSU.....	403
	2.6.9.3.6	init_target_list CSU	403
	2.6.9.3.7	set_target_list CSU.....	403
	2.6.9.3.8	targeting_set_fire_at_pointair CSU.....	404
	2.6.9.3.9	targeting_set_hold_fire CSU.....	404
	2.6.9.3.10	targeting_set_fire_at_will CSU	404
	2.6.9.3.11	targeting_set_parameters CSU.....	404
	2.6.9.3.12	TARGET_ITEM CSU	405
	2.6.9.3.13	init_target_items CSU	405
	2.6.9.3.14	compare_targets CSU.....	405
	2.6.9.3.15	target_priority CSU	405
	2.6.9.3.16	target_type_ok CSU	406
	2.6.9.3.17	target_in_position CSU.....	407
	2.6.9.3.18	select_weapon_priority_list CSU	408
	2.6.9.3.19	update_target_list CSU	408
	2.6.9.3.20	choose_target_and_weapon CSU	409
	2.6.9.3.21	targeting_tick CSU.....	409
2.6.9.4		targeting.h CSU	410
2.6.9.5		gunner.c CSC.....	411
	2.6.9.5.1	acquire_target CSU	411
	2.6.9.5.2	track_target CSU	412
	2.6.9.5.3	point_weapon_at_target CSU.....	412
	2.6.9.5.4	scan_weapon CSU.....	413
	2.6.9.5.5	shoot_target CSU	413
	2.6.9.5.6	gunner_round_flying CSU.....	414
	2.6.9.5.7	gunner_tick CSU.....	414
2.6.9.6		turret.c CSC.....	415
	2.6.9.6.1	create_turn CSU	415

	2.6.9.6.2	destroy_turret CSU	416
	2.6.9.6.3	turret_show CSU.....	416
	2.6.9.6.4	turret_slew CSU.....	416
	2.6.9.6.5	turret_point_at_target CSU.....	417
	2.6.9.6.6	turret_scan CSU.....	418
	2.6.9.6.7	turret_muzzle_position_in_ world_coordinates CSU.....	418
	2.6.9.6.8	turret_set_scan_parms CSU	418
	2.6.9.6.9	turret_interest_dir CSU.....	419
	2.6.9.6.10	turret_firepower_kill CSU	419
2.6.9.7	turret.h CSU		419
2.6.9.8	weapons.c CSC		420
	2.6.9.8.1	weapon_state_to_string CSU.....	420
	2.6.9.8.2	create_weapon_systems CSU	421
	2.6.9.8.3	destroy_weapon_systems CSU	422
	2.6.9.8.4	weapon_configure CSU.....	422
	2.6.9.8.5	weapon_systems_show CSU.....	422
	2.6.9.8.6	weapon_show CSU.....	423
	2.6.9.8.7	weapon_priority_list_show CSU	423
	2.6.9.8.8	weapon_systems_rearm CSU	423
	2.6.9.8.9	weapon_systems_checkpoint CSU.....	424
	2.6.9.8.10	weapon_systems_reinit CSU	424
	2.6.9.8.11	clear_weapons_status CSU	424
	2.6.9.8.12	add_weapons_status CSU	424
	2.6.9.8.13	initialize_weapon_priority_list CSU.....	425
	2.6.9.8.14	weapon_select CSU.....	425
	2.6.9.8.15	weapon_deselect CSU.....	425
	2.6.9.8.16	weapon_load CSU	426
	2.6.9.8.17	weapon_unload CSU	426
	2.6.9.8.18	muzzle_position_in_world_ coordinates CSU	426
	2.6.9.8.19	radiate_target CSU	427
	2.6.9.8.20	get_relative_vehicle_agility CSU.....	427
	2.6.9.8.21	fire_weapon_at_target CSU.....	428
	2.6.9.8.22	fly_round CSU	429
	2.6.9.8.23	impact_weapon CSU	430
	2.6.9.8.24	generate_weapon_hit CSU.....	430
	2.6.9.8.25	generate_weapon_miss CSU	431
2.6.9.9	weapons.h CSU.....		431
2.6.9.10	loader.c CSC.....		432
	2.6.9.10.1	loader_tick CSU.....	433

2.7	REMOTE VEHICLES CSC	434
2.7.1	remote.c CSC.....	435
2.7.1.1	create_remote CSU.....	435
2.7.1.2	remote_init_vars CSU	436
2.7.1.3	remote_go_away CSU.....	436
2.7.1.4	remote_show CSU.....	436
2.7.1.5	remote_start_ticking CSU.....	437
2.7.1.6	remote_fill_in_appearance_data CSU.....	437
2.7.1.7	remote_fill_echelon_data CSU	438
2.7.1.8	remote_fill_in_position_data CSU	438
2.7.1.9	remote_next_road_point CSU	438
2.7.1.10	remote_start_being_watched CSU	439
2.7.1.11	remote_stop_being_watched CSU	439
2.7.1.12	remote_new_appearance CSU.....	440
2.7.1.13	remote_tick CSU	441
2.7.1.14	send_stealth_gone_msg CSU.....	441
2.7.1.15	remote_deactivate CSU.....	442
2.7.1.16	remote_change_stealth_controlling_port CSU.....	442
2.7.1.17	create_remote_vehicle CSU.....	443
2.8	UNITS CSC.....	444
2.8.1	Command and Control CSC.....	444
2.8.1.1	commander.c CSC.....	445
2.8.1.1.1	create_commander CSU.....	446
2.8.1.1.2	destroy_commander CSU	446
2.8.1.1.3	commander_state_string CSU	446
2.8.1.1.4	commander_show CSU	447
2.8.1.1.5	commander_get_mission_status CSU.....	447
2.8.1.1.6	commander_executing_order CSU.....	447
2.8.1.1.7	commander_execute_overlay CSU.....	448
2.8.1.1.8	commander_cancel_overlay CSU	448
2.8.1.1.9	commander_tick CSU	449
2.8.1.1.10	get_a_leader CSU	449
2.8.1.1.11	commander_halt CSU	450
2.8.1.1.12	commander_set_mission_speed CSU.....	450
2.8.1.1.13	commander_set_mission_direction CSU.....	451
2.8.1.1.14	commander_change_speed CSU	451
2.8.1.1.15	commander_set_mission_ formation CSU	451

2.8.1.1.16	commander_change_formation CSU.....	452
2.8.1.1.17	commander_follow_vehicle CSU	452
2.8.1.1.18	commander_simulator_in_ command CSU	453
2.8.1.1.19	commander_goto_point CSU.....	453
2.8.1.1.20	commander_resume_mission CSU.....	454
2.8.1.1.21	commander_face_direction CSU	454
2.8.1.1.22	commander_inferior_changed_ status CSU	455
2.8.1.1.23	commander_note_leader_state CSU.....	455
2.8.1.1.24	commander_restore_leader_state CSU.....	455
2.8.1.1.25	commander_land CSU.....	456
2.8.1.1.26	commander_change_altitude CSU ...	456
2.8.1.1.27	commander_attack CSU.....	456
2.8.1.2	commander.h CSU.....	457
2.8.1.3	composite.c CSC.....	457
2.8.1.3.1	create_composite CSU.....	459
2.8.1.3.2	composite_go_away CSU.....	460
2.8.1.3.3	composite_show CSU.....	461
2.8.1.3.4	composite_set_superior CSU.....	461
2.8.1.3.5	composite_add_inferior_composite CSU.....	462
2.8.1.3.6	composite_remove_inferior_ composite CSU.....	462
2.8.1.3.7	composite_add_inferior_vehicle CSU.....	462
2.8.1.3.8	composite_remove_inferior_ vehicle CSU.....	463
2.8.1.3.9	composite_add_member_vehicle CSU.....	463
2.8.1.3.10	composite_remove_member_ vehicle CSU.....	463
2.8.1.3.11	composite_start_ticking CSU.....	464
2.8.1.3.12	composite_fill_in_echelon_data CSU.....	464
2.8.1.3.13	composite_fill_in_position_data CSU.....	464
2.8.1.3.14	composite_fill_in_appearance_data CSU.....	465
2.8.1.3.15	composite_find_living_vehicle CSU.....	465

2.8.1.3.16	composite_distinguished_member CSU.....	465
2.8.1.3.17	composite_check_unit_strength CSU.....	466
2.8.1.3.18	composite_send_unit_strength_ message CSU.....	466
2.8.1.3.19	composite_tick CSU	467
2.8.1.3.20	composite_pro_sim CSU.....	467
2.8.1.3.21	composite_indirect_fire CSU	468
2.8.1.3.22	composite_get_composite_ bumpers CSU.....	468
2.8.1.3.23	composite_set_recursive_ids CSU....	468
2.8.1.3.24	composite_set_sbx_ids CSU	469
2.8.1.3.25	composite_assume_formation_ internal CSU	469
2.8.1.3.26	composite_assume_formation CSU.....	470
2.8.1.3.27	composite_reassign_current_ formation CSU	470
2.8.1.3.28	composite_teleport_to_station CSU.....	470
2.8.1.3.29	composite_find_vehicle_for_ stealth CSU	471
2.8.1.3.30	unit_send_pae_data CSU.....	471
2.8.1.3.31	composite_set_targeting_ parameters CSU	472
2.8.1.3.32	composite_fake_resupply CSU.....	472
2.8.1.3.33	composite_generate_status_report CSU.....	472
2.8.1.3.34	composite_remove_vehicles CSU.....	473
2.8.1.3.35	composite_executing_order CSU	473
2.8.1.3.36	composite_note_leader_state CSU	474
2.8.1.3.37	composite_inferior_changed_ status CSU	474
2.8.1.3.38	composite_note_member_vehicle_ has_died CSU	474
2.8.1.3.39	composite_rejoin_unit CSU.....	475
2.8.1.3.40	composite_execute_overlay CSU	475
2.8.1.3.41	composite_readjust_overlay CSU....	476
2.8.1.3.42	composite_cancel_overlay CSU.....	476
2.8.1.3.43	composite_halt CSU	476
2.8.1.3.44	composite_change_speed CSU.....	477
2.8.1.3.45	composite_follow_vehicle CSU.....	477

	2.8.1.3.46	composite_simulator_in_command CSU.....	477
	2.8.1.3.47	composite_change_formation CSU.....	478
	2.8.1.3.48	composite_goto_point CSU.....	478
	2.8.1.3.49	composite_resume_mission CSU.....	478
	2.8.1.3.50	composite_face_direction CSU.....	479
	2.8.1.3.51	composite_attack CSU.....	479
	2.8.1.3.52	composite_land CSU.....	479
	2.8.1.3.53	composite_change_altitude CSU.....	480
2.8.1.4	formation.c	CSC.....	480
	2.8.1.4.1	get_formation_layout CSU.....	481
	2.8.1.4.2	get_turret CSU.....	482
	2.8.1.4.3	spread_increment CSU.....	482
	2.8.1.4.4	compute_who_follows_whom CSU.....	483
	2.8.1.4.5	int_dist CSU.....	483
	2.8.1.4.6	closest_to_a_leader CSU.....	484
	2.8.1.4.7	closest_leader CSU.....	484
	2.8.1.4.8	make_leader CSU.....	485
	2.8.1.4.9	print_lists CSU.....	485
	2.8.1.4.10	get_station_info CSU.....	485
	2.8.1.4.11	cmp_form_entries CSU.....	486
	2.8.1.4.12	sort_form_db CSU.....	486
2.8.2	Reports	CSC.....	487
	2.8.2.1	reporter.c	CSC.....487
	2.8.2.1.1	create_reporter CSU.....	488
	2.8.2.1.2	destroy_reporter CSU.....	488
	2.8.2.1.3	reporter_remove_vehicles CSU.....	489
	2.8.2.1.4	reporter_show CSU.....	489
	2.8.2.1.5	heading_to_compass CSU.....	490
	2.8.2.1.6	update_cluster_heading CSU.....	490
	2.8.2.1.7	show_vehicle_clusters CSU.....	491
	2.8.2.1.8	show_vehicle_cluster CSU.....	491
	2.8.2.1.9	show_vehicle_id_list CSU.....	491
	2.8.2.1.10	destroy_vehicle_clusters CSU.....	491
	2.8.2.1.11	destroy_vehicle_cluster CSU.....	492
	2.8.2.1.12	destroy_veh_id_list CSU.....	492
	2.8.2.1.13	destroy_veh_id_item CSU.....	492
	2.8.2.1.14	destroy_removed_vehicles CSU.....	493
	2.8.2.1.15	destroy_rem_veh_id_item CSU.....	493
	2.8.2.1.16	destroy_shellings CSU.....	493
	2.8.2.1.17	destroy_shell_summary CSU.....	494
	2.8.2.1.18	vehicle_list_append CSU.....	494

2.8.2.1.19	create_new_cluster CSU	494
2.8.2.1.20	add_vehicle_to_cluster CSU	495
2.8.2.1.21	splice_out_vehicle_cluster CSU	495
2.8.2.1.22	update_cluster_com CSU	496
2.8.2.1.23	need_decluster CSU	496
2.8.2.1.24	cluster_vehicle CSU	496
2.8.2.1.25	cluster_from_vehicle CSU	497
2.8.2.1.26	remove_clustered_vehicle CSU	497
2.8.2.1.27	vehicle_reappeared CSU	498
2.8.2.1.28	show_removed_vehicles CSU	498
2.8.2.1.29	add_removed_vehicle CSU	498
2.8.2.1.30	remove_removed_vehicle CSU	499
2.8.2.1.31	remove_dead_clustered_vehicles CSU	499
2.8.2.1.32	update_clusters CSU	499
2.8.2.1.33	shell_type_to_string CSU	500
2.8.2.1.34	ammo_to_shell_type CSU	500
2.8.2.1.35	show_shelling_clusters CSU	501
2.8.2.1.36	create_new_shelling CSU	501
2.8.2.1.37	add_shelling_to_cluster CSU	502
2.8.2.1.38	remove_shelling_cluster CSU	502
2.8.2.1.39	cluster_shell CSU	502
2.8.2.1.40	shell_report_needed CSU	503
2.8.2.1.41	reporter_tick CSU	503
2.8.2.1.42	update_reporter_vehicles CSU	504
2.8.2.1.43	issue_reports CSU	504
2.8.2.1.44	report_ivis_network CSU	505
2.8.2.1.45	report_cluster_vehicle_type CSU	505
2.8.2.1.46	send_contact_report CSU	506
2.8.2.1.47	send_spot_report CSU	506
2.8.2.1.48	send_shelling_report CSU	507
2.8.2.2	reporter.h CSU	508
2.8.2.3	spotter.c CSC	510
2.8.2.3.1	create_spotter CSU	510
2.8.2.3.2	destroy_spotter CSU	511
2.8.2.3.3	allocate_spotter_tables CSU	511
2.8.2.3.4	deallocate_spotter_tables CSU	511
2.8.2.3.5	spotter_init_table CSU	512
2.8.2.3.6	spotter_remove_vehicles CSU	512
2.8.2.3.7	spotter_tick CSU	512
2.8.2.3.8	spotter_sum_inferior_composite_ tables CSU	513
2.8.2.3.9	add_into_spotter_table CSU	513

	2.8.2.3.10	major_spotter_increase CSU.....	514
	2.8.2.3.11	spotter_save_weight CSU.....	514
	2.8.2.3.12	spotter_show CSU.....	515
	2.8.2.3.13	spotter_show_table CSU.....	515
	2.8.2.4	spotter.h CSU	515
2.8.3	Situation Assessment CSC.....		516
	2.8.3.1	tactical_state.h CSU.....	516
2.9	SAF OBJECTS CSC		517
2.9.1	SAF Object Structures CSC		517
	2.9.1.1	safobj.h CSU	517
	2.9.1.2	veh_storage.h CSU	518
2.9.2	SAF Object Operations CSC.....		530
	2.9.2.1	entity.c CSC	530
	2.9.2.1.1	create_entity CSU	531
	2.9.2.1.2	destroy_entity CSU	531
	2.9.2.1.3	force_id_to_string CSU.....	532
	2.9.2.1.4	entity_show CSU	532
	2.9.2.1.5	entity_fill_in_position_data CSU.....	532
	2.9.2.1.6	entity_fill_in_appearance_data CSU.....	533
	2.9.2.2	vehicle.c CSC.....	533
	2.9.2.2.1	create_vehicle CSU	534
	2.9.2.2.2	destroy_vehicle CSU	534
	2.9.2.2.3	print_vehicle_marking CSU	534
	2.9.2.2.4	vehicle_show CSU.....	535
	2.9.2.2.5	vehicle_fill_in_position_data CSU....	535
	2.9.2.2.6	vehicle_fill_in_appearance_data CSU.....	535
	2.9.2.2.7	vehicle_planar_distance_squared CSU.....	536
	2.9.2.2.8	vehicle_deactivate CSU.....	536
2.9.3	Iterators CSC.....		536
	2.9.3.1	iterator.c CSC.....	536
	2.9.3.1.1	vehicle_iterator_reset CSU.....	537
	2.9.3.1.2	vehicle_iterator_next CSU	537
	2.9.3.1.3	vehicle_iterator_once_next CSU	538
	2.9.3.1.4	vehicle_manager_print CSU	538
	2.9.3.1.5	vehicle_manager_count CSU	539
	2.9.3.1.6	vehicle_manager_count_force CSU.....	539
	2.9.3.1.7	vehicle_manager_count_sites CSU.....	539
	2.9.3.1.8	vehicle_manager_count_hosts CSU.....	540

	2.9.3.1.9	host_local_vehicle_predicate CSU	540
	2.9.3.1.10	init_grid_tables CSU.....	540
	2.9.3.1.11	init_grid_entry_list CSU	541
	2.9.3.1.12	update_grid_entry_list CSU	541
	2.9.3.1.13	destroy_grid_entry_list CSU	542
	2.9.3.1.14	remove_grid_entry CSU	542
	2.9.3.1.15	insert_grid_entry CSU.....	542
	2.9.3.2	iterator.h CSU	542
2.10	OPORDERS CSC.....		545
2.10.1	Combat Instruction Sets CSC		546
2.10.1.1	cis.c CSC.....		546
	2.10.1.1.1	lookup_cis_situational CSU	547
	2.10.1.1.2	lookup_cis_resumable CSU	547
	2.10.1.1.3	lookup_cis_default_speed CSU	548
	2.10.1.1.4	lookup_cis_communication_string CSU.....	548
	2.10.1.1.5	lookup_cis_movement_method CSU.....	549
	2.10.1.1.6	lookup_cis_enabled_predicates CSU.....	549
	2.10.1.1.7	lookup_cis_formation CSU.....	549
	2.10.1.1.8	create_cis CSU	550
	2.10.1.1.9	destroy_cis CSU	550
	2.10.1.1.10	cis_show CSU.....	551
	2.10.1.1.11	push_cis CSU	551
	2.10.1.1.12	pop_cis CSU.....	551
	2.10.1.1.13	pop_resumable_cis CSU.....	552
	2.10.1.1.14	clear_cis_stack CSU	552
	2.10.1.1.15	adjust_behaviors CSU.....	552
	2.10.1.1.16	activate_new_cis CSU.....	553
	2.10.1.1.17	cm_to_passage_string CSU.....	554
	2.10.1.1.18	fill_generic_cis_report CSU	554
	2.10.1.1.19	change_movement_and_report CSU.....	555
	2.10.1.2	cis.h CSU	555
	2.10.1.3	predicates.c CSC.....	556
	2.10.1.3.1	create_predicates CSU.....	556
	2.10.1.3.2	destroy_predicates CSU.....	557
	2.10.1.3.3	hasty_attack_needed CSU.....	557
	2.10.1.3.4	air_raid_happening CSU	558
	2.10.1.3.5	air_raid_over CSU	558
	2.10.1.3.6	action_drill_needed CSU.....	558
	2.10.1.3.7	predicate_impact_hook CSU.....	559

	2.10.1.3.8	enemy_disposition CSU.....	559
	2.10.1.3.9	predicates_show CSU	560
	2.10.1.3.10	reset_predicates CSU	560
	2.10.1.3.11	enable_predicate CSU	560
	2.10.1.3.12	enable_predicates CSU	561
	2.10.1.3.13	check_predicates CSU.....	561
	2.10.1.4	predicates.h CSU.....	562
2.10.2	Control Measure CSC.....		563
	2.10.2.1	cm.c CSC	563
	2.10.2.1.1	copy_float_point_list_to_real_	
		point_list CSU.....	564
	2.10.2.1.2	get_overlay CSU	564
	2.10.2.1.3	get_control_measure CSU	565
	2.10.2.1.4	cm_find_overlay_list_item CSU	565
	2.10.2.1.5	cm_find_overlay CSU.....	566
	2.10.2.1.6	cm_find_cm_list_item CSU.....	566
	2.10.2.1.7	cm_find_cm CSU	567
	2.10.2.1.8	cm_add_point CSU	567
	2.10.2.1.9	cm_add_area CSU	568
	2.10.2.1.10	cm_add_zone CSU.....	568
	2.10.2.1.11	cm_add_line CSU.....	569
	2.10.2.1.12	cm_add_route CSU	569
	2.10.2.1.13	cm_delete_overlay CSU.....	570
	2.10.2.1.14	cm_delete_cm CSU	570
	2.10.2.1.15	cm_delete_cm_from_overlay	
		CSU.....	570
	2.10.2.1.16	cm_applies_to_units CSU.....	571
	2.10.2.1.17	cm_get_cm_from_overlay CSU.....	571
	2.10.2.1.18	is_cm_executed_in_list CSU	572
	2.10.2.1.19	cm_recompute_cm_from_overlay	
		CSU.....	572
	2.10.2.1.20	cm_execute_overlay CSU	573
	2.10.2.1.21	add_executing_unit CSU.....	573
	2.10.2.1.22	remove_executing_unit CSU	574
	2.10.2.1.23	cm_force_overlay_recalculation	
		CSU.....	574
	2.10.2.1.24	destroy_cm_data CSU.....	574
	2.10.2.1.25	destroy_point_cm CSU.....	575
	2.10.2.1.26	destroy_area_cm CSU.....	575
	2.10.2.1.27	destroy_zone_cm CSU	575
	2.10.2.1.28	destroy_line_cm CSU	576
	2.10.2.1.29	destroy_cm CSU	576
	2.10.2.1.30	destroy_cm_list_item CSU.....	576
	2.10.2.1.31	destroy_cm_list CSU	577

	2.10.2.1.32	destroy_overlay CSU.....	577
	2.10.2.1.33	destroy_overlay_list_item CSU	577
	2.10.2.1.34	show_overlays CSU.....	578
	2.10.2.1.35	show_overlay CSU	578
	2.10.2.1.36	show_control_measures CSU	578
	2.10.2.1.37	show_control_measure CSU	579
	2.10.2.1.38	show_point_list CSU.....	579
	2.10.2.1.39	show_point_cm CSU.....	579
	2.10.2.1.40	show_area_cm CSU	579
	2.10.2.1.41	show_zone_cm CSU	580
	2.10.2.1.42	show_line_cm CSU.....	580
	2.10.2.1.43	show_applies_to CSU.....	580
2.10.2.2		cm.h CSU.....	581
2.10.2.3		navigator.c CSC.....	584
	2.10.2.3.1	create_navigator CSU.....	584
	2.10.2.3.2	destroy_navigator CSU.....	585
	2.10.2.3.3	navigator_show CSU.....	585
	2.10.2.3.4	navigator_set_overlay CSU.....	585
	2.10.2.3.5	navigator_reset_overlay CSU.....	586
	2.10.2.3.6	navigator_tick CSU	586
	2.10.2.3.7	navigator_passed_point CSU.....	587
	2.10.2.3.8	navigator_passed_line CSU.....	587
	2.10.2.3.9	navigator_entered_area CSU.....	587
	2.10.2.3.10	navigator_left_area CSU	588
	2.10.2.3.11	navigator_entered_zone CSU.....	588
	2.10.2.3.12	navigator_left_zone CSU.....	589
	2.10.2.3.13	cm_intersection CSU	589
	2.10.2.3.14	cm_poly_intersection CSU.....	590
	2.10.2.3.15	cm_get_center_of_mass CSU	590
	2.10.2.3.16	cm_point_inside_polygon CSU	590
	2.10.2.3.17	cm_count_intersections CSU.....	591
	2.10.2.3.18	cm_point_line_intersection CSU	591
	2.10.2.3.19	navigator_generate_report CSU	592
2.10.2.4		route.c CSC.....	593
	2.10.2.4.1	routepoint_distance CSU.....	593
	2.10.2.4.2	append_routepoint CSU.....	594
	2.10.2.4.3	create_routepoint CSU.....	594
	2.10.2.4.4	destroy_route CSU.....	594
	2.10.2.4.5	destroy_routepoint CSU.....	595
	2.10.2.4.6	expand_road_route CSU	595
	2.10.2.4.7	generate_route_from_route_msg CSU	596
	2.10.2.4.8	generate_in_place_route CSU	596

	2.10.2.4.9	eliminate_duplicates CSU	597
	2.10.2.4.10	exaggerate_bridges CSU	597
	2.10.2.4.11	avoid_water_to_next_point CSU	597
	2.10.2.4.12	fragment_route CSU	598
	2.10.2.4.13	re_assign_routepoints CSU	598
	2.10.2.4.14	show_route CSU	598
	2.10.2.4.15	show_route_points CSU	598
	2.10.2.4.16	print_routepoint CSU	599
	2.10.2.4.17	print_route CSU	599
	2.10.2.5	route.h CSU	600
2.11	CREATE CSC		601
2.11.1	create.c CSC		601
	2.11.1.1	make_create_list CSU	602
	2.11.1.2	create_unit CSU	602
2.12	TERRAIN CSC		604
2.12.1	libquad CSC		605
	2.12.1.1	terrain.c CSC	605
	2.12.1.1.1	read_quadtree_database CSU	605
	2.12.1.1.2	read_terrain_files CSU	606
	2.12.1.1.3	read_quadtree CSU	606
	2.12.1.1.4	read_feature CSU	606
	2.12.1.1.5	allocate_points CSU	607
	2.12.1.1.6	warn_about_point_space CSU	607
	2.12.1.2	rivers.c CSC	607
	2.12.1.2.1	create_rivers CSU	608
	2.12.1.2.2	print_river_segment CSU	608
	2.12.1.2.3	create_river_ints CSU	608
	2.12.1.2.4	print_river_intersection CSU	608
	2.12.1.3	bridges.c CSC	609
	2.12.1.3.1	create_bridges CSU	609
	2.12.1.3.2	print_bridge CSU	609
	2.12.1.3.3	create_lakes CSU	610
	2.12.1.3.4	print_lake CSU	610
	2.12.1.4	water_check.c CSC	610
	2.12.1.4.1	get_water_indicies CSU	611
	2.12.1.4.2	free_water_list CSU	611
	2.12.1.4.3	any_wide_segment_thru_water CSU	612
	2.12.1.4.4	water_check CSU	612
	2.12.1.4.5	segment_thru_water CSU	613
	2.12.1.4.6	segment_thru_river CSU	613
	2.12.1.4.7	segment_thru_lake CSU	614

	2.12.1.4.8	all_wide_segments_thru_water CSU.....	615
	2.12.1.4.9	water_thru CSU	615
	2.12.1.4.10	sort_water CSU	616
	2.12.1.4.11	water_segments_thru CSU.....	616
	2.12.1.4.12	lakes_thru CSU.....	617
2.12.1.5	search.c CSC		618
	2.12.1.5.1	find_a_quad CSU	618
	2.12.1.5.2	get_quad_nodes CSU.....	619
	2.12.1.5.3	get_quad_nodes_internal CSU	619
	2.12.1.5.4	valid_quad CSU.....	620
2.12.1.6	print.c CSC		620
	2.12.1.6.1	print_quad_node CSU.....	620
2.12.1.7	list.c CSC		621
	2.12.1.7.1	cons CSU	621
	2.12.1.7.2	ncons CSU.....	622
	2.12.1.7.3	push CSU	622
	2.12.1.7.4	free_list CSU	623
	2.12.1.7.5	remove_duplicates CSU.....	623
	2.12.1.7.6	member CSU	623
	2.12.1.7.7	delete CSU.....	624
	2.12.1.7.8	nthcdr CSU	624
	2.12.1.7.9	nconc CSU.....	624
	2.12.1.7.10	last_item CSU	625
	2.12.1.7.11	last_item_butlast CSU	625
	2.12.1.7.12	delete_last_item CSU.....	625
	2.12.1.7.13	set_xor CSU	626
	2.12.1.7.14	copy_list CSU	626
	2.12.1.7.15	reverse CSU	627
	2.12.1.7.16	length CSU.....	627
2.12.1.8	water_avoidance.c CSC.....		627
	2.12.1.8.1	find_route CSU.....	628
	2.12.1.8.2	find_route_core CSU.....	629
	2.12.1.8.3	follow_water_segments CSU.....	630
	2.12.1.8.4	find_suitable_crossing_route CSU.....	630
2.12.1.9	vector_2d.c CSC.....		631
	2.12.1.9.1	make_vector_2d CSU	632
	2.12.1.9.2	vec_normalize_2d CSU	632
	2.12.1.9.3	vec_rotate_2d CSU	632
	2.12.1.9.4	vec_distance_2d CSU	633
2.12.1.10	extend_crossings.c CSC		633
	2.12.1.10.1	extend_crossing CSU.....	633

	2.12.1.10.2	extend_intersection CSU	634
	2.12.1.10.3	extend_bridge CSU	635
	2.12.1.10.4	extend_segment CSU	635
	2.12.1.10.5	find_segment_cross_points CSU	636
2.12.1.11	water_utilities.c	CSC	637
	2.12.1.11.1	distance CSU	637
	2.12.1.11.2	intersection_direction CSU	638
	2.12.1.11.3	find_direction_at_crossing CSU	638
	2.12.1.11.4	normalize_and_rotate CSU	639
	2.12.1.11.5	find_first_vector CSU	639
	2.12.1.11.6	vector_is_first CSU	640
	2.12.1.11.7	find_closer_crossing CSU	640
	2.12.1.11.8	find_next_point CSU	641
	2.12.1.11.9	crossing_location CSU	641
	2.12.1.11.10	relax_points CSU	641
	2.12.1.11.11	relax_points_aux CSU	642
	2.12.1.11.12	final_relax_points CSU	643
	2.12.1.11.13	get_quads_in_region CSU	643
	2.12.1.11.14	get_quads_in_rt_point_region CSU	644
	2.12.1.11.15	get_quads_in_vector_region CSU ...	644
	2.12.1.11.16	distance_around_path CSU	644
	2.12.1.11.17	push_vector_2d_on_points CSU	645
	2.12.1.11.18	push_vector_2d_on_route CSU	645
	2.12.1.11.19	append_vector_2d_on_route CSU ...	646
	2.12.1.11.20	free_crossings CSU	646
	2.12.1.11.21	rt_point_to_vector CSU	646
	2.12.1.11.22	print_route_list CSU	647
	2.12.1.11.23	print_vector_list CSU	647
2.12.1.12	skirt_water.c	CSC	647
	2.12.1.12.1	skirt_river CSU	648
	2.12.1.12.2	find_river_points CSU	648
	2.12.1.12.3	align_points CSU	649
	2.12.1.12.4	offset_points CSU	649
	2.12.1.12.5	offset_point CSU	650
	2.12.1.12.6	prune_to_point CSU	650
	2.12.1.12.7	skirt_lake CSU	651
	2.12.1.12.8	follow_lake_around CSU	651
	2.12.1.12.9	skirt_river_bend CSU	652
	2.12.1.12.10	find_river_bend_points CSU	653
2.12.1.13	minimum_clip.c	CSC	653
	2.12.1.13.1	minimum_clip CSU	654
	2.12.1.13.2	left_column	654

2.12.1.13.3	top_left_corner CSU.....	655
2.12.1.13.4	left_bottom_region CSU	655
2.12.1.13.5	left_edge CSU	656
2.12.1.13.6	p2_bottom CSU	656
2.12.1.13.7	center_column CSU.....	657
2.12.1.14	intersection.c CSC	657
2.12.1.14.1	seg_intersection CSU.....	658
2.12.1.14.2	possible_intersection CSU	658
2.12.1.14.3	point_inside_polygon CSU	659
2.12.1.14.4	count_intersections CSU	659
2.12.1.14.5	segment_inside_polygon CSU	660
2.12.1.14.6	point_line_intersection CSU	660
2.12.1.14.7	point_segment_intersection CSU.....	661
2.12.1.14.8	line_intersection CSU.....	661
2.12.1.14.9	open_line_intersection CSU	662
2.12.1.14.10	line_intersection_core CSU	663
2.12.1.15	roads.c CSC	663
2.12.1.15.1	create_roads CSU	664
2.12.1.15.2	print_road_segment CSU	664
2.12.1.15.3	create_road_ints CSU.....	664
2.12.1.15.4	print_road_intersection CSU.....	665
2.12.1.15.5	next_road_point CSU.....	665
2.12.1.16	trees.c CSC	666
2.12.1.16.1	create_trees CSU	666
2.12.1.16.2	print_tree CSU.....	667
2.12.1.16.3	create_tree_canopies CSU.....	667
2.12.1.16.4	print_tree_canopy CSU.....	667
2.12.1.16.5	create_buildings CSU.....	668
2.12.1.16.6	print_building CSU	668
2.12.1.16.7	create_contours CSU	668
2.12.1.16.8	find_extra_contour_info CSU	669
2.12.1.17	building_check.c CSC.....	669
2.12.1.17.1	buildings_thru CSU.....	670
2.12.1.17.2	point_in_building CSU	670
2.12.1.17.3	find_closest_building CSU	671
2.12.1.17.4	get_building_indecies CSU	671
2.12.1.17.5	expand_points CSU.....	672
2.12.1.18	libquad.h CSU.....	673
2.13	GLOBAL CSC	681
2.13.1	Global Variables CSC.....	681
2.13.1.1	globals.h CSU.....	681
2.13.2	External Functions CSC.....	681
3.2.1	prototypes.h CSU	682

2.13.3	Macros and Constants CSC.....	682
2.13.3.1	phantom.h CSU	682
2.13.3.2	abs.h CSU	683
2.13.3.3	longpt.h CSU.....	683
2.13.3.4	mass_std.c.h CSU	683
2.13.3.5	minmax.h CSU	684
2.13.3.6	sim_undef.h CSU.....	684
2.13.3.7	tolerance.h CSU.....	684
2.14	UTILITIES CSC.....	685
2.14.1	Utilities CSC.....	685
2.14.1.1	dispatch.c CSC.....	686
2.14.1.1.1	fill_in_appearance_data CSU	686
2.14.1.1.2	fill_in_echelon_data CSU	687
2.14.1.1.3	fill_in_position_data CSU.....	687
2.14.1.1.4	show CSU	688
2.14.1.1.5	remove_vehicles CSU	688
2.14.1.1.6	go_away CSU	688
2.14.1.1.7	set_targeting_parameters CSU.....	689
2.14.1.1.8	execute_overlay CSU	689
2.14.1.1.9	readjust_overlay CSU	690
2.14.1.1.10	cancel_overlay CSU	690
2.14.1.1.11	generate_status_report CSU	691
2.14.1.1.12	fake_resupply CSU	691
2.14.1.2	misc.c CSC.....	692
2.14.1.2.1	get_soil_type CSU	692
2.14.1.2.2	tdb_get_gl CSU	693
2.14.1.2.3	tdb_get_zl CSU.....	693
2.14.1.2.4	report_error_from_tdb_once CSU	694
2.14.1.2.5	copy_xy_on_tdb CSU.....	694
2.14.1.2.6	r2 CSU	694
2.14.1.2.7	noa_damp CSU.....	695
2.14.1.2.8	dotv CSU.....	695
2.14.1.2.9	diffv CSU	695
2.14.1.2.10	within_delta CSU	696
2.14.1.2.11	ft_int CSU.....	696
2.14.1.2.12	ft_float CSU	697
2.14.1.2.13	ft_symbol CSU.....	697
2.14.1.2.14	ft_table CSU	697
2.14.1.2.15	ft_untagged_table CSU.....	698
2.14.2	libheap CSC.....	698
2.14.2.1	alloc.c CSC.....	699
2.14.2.1.1	heap_allocate CSU	700
2.14.2.1.2	heap_calloc CSU	700

	2.14.2.1.3	heap_allocate_distributed CSU	701
	2.14.2.1.4	heap_deallocate CSU	701
	2.14.2.1.5	init_free_list CSU	701
	2.14.2.1.6	collect CSU	702
	2.14.2.1.7	break_block CSU	702
	2.14.2.1.8	remove_from_free_list CSU	702
	2.14.2.1.9	add_to_free_list CSU	702
	2.14.2.1.10	heap_print_statistics CSU	703
	2.14.2.1.11	print_heap_table CSU	703
	2.14.2.1.12	heap_verify CSU	703
	2.14.2.1.13	heap_inconsistent_action CSU	704
	2.14.2.1.14	heap_hang CSU	704
	2.14.2.1.15	print_block CSU	704
	2.14.2.1.16	do_heap_verify CSU	704
	2.14.2.1.17	do_heap_statistics CSU	705
	2.14.2.1.18	do_heap_collect CSU	705
	2.14.2.1.19	apply_to_blocks_of_size CSU	705
	2.14.2.2	create.c CSC	706
	2.14.2.2.1	initialize_heap CSU	706
	2.14.2.2.2	heap_create CSU	706
	2.14.2.2.3	heap_destroy CSU	707
	2.14.2.3	libheap.h CSU	707
2.14.3	libutil CSC		708
	2.14.3.1	args.c CSC	708
	2.14.3.1.1	proc_switches CSU	708
	2.14.3.2	checksum.c CSC	709
	2.14.3.2.1	checksum CSU	709
	2.14.3.3	time.c CSC	710
	2.14.3.3.1	init_clocks CSU	710
	2.14.3.3.2	get_millisecond_time CSU	710
	2.14.3.4	bitfield.c CSC	711
	2.14.3.4.1	allocate_bitfield CSU	711
	2.14.3.4.2	deallocate_bitfield CSU	712
	2.14.3.4.3	set_bit CSU	712
	2.14.3.4.4	clear_bit CSU	712
	2.14.3.4.5	copy_bitfield CSU	712
	2.14.3.4.6	clear_bitfield CSU	713
	2.14.3.4.7	set_bitfield CSU	713
	2.14.3.4.8	or_bitfield CSU	713
	2.14.3.4.9	and_bitfield CSU	713
	2.14.3.4.10	count_bits CSU	714
	2.14.3.5	math.c CSC	715
	2.14.3.5.1	print_matrix CSU	716

2.14.3.5.2	print_vector CSU.....	716
2.14.3.5.3	make_vector_from_angle_ magnitude CSU.....	716
2.14.3.5.4	angle_clip CSU.....	716
2.14.3.5.5	clip_angle_positive CSU.....	717
2.14.3.5.6	clip_angle_negative CSU.....	717
2.14.3.5.7	angle_between_vectors CSU	717
2.14.3.5.8	interior_angle_between_vectors CSU.....	718
2.14.3.5.9	which_side CSU	718
2.14.3.5.10	range_squared CSU.....	718
2.14.3.5.11	round_real_to_int CSU.....	719
2.14.3.5.12	vector_z_rotate CSU.....	719
2.14.3.5.13	fvec_to_rvec CSU.....	719
2.14.3.5.14	rvec_to_fvec CSU.....	720
2.14.3.5.15	rmat_to_fmat CSU	720
2.14.3.5.16	fmat_to_rmat CSU	720
2.14.3.5.17	fvec_copy CSU.....	720
2.14.3.5.18	copy_matrix_row_to_vector CSU ...	721
2.14.3.5.19	vec2_print CSU	721
2.14.3.5.20	vec2_init CSU	721
2.14.3.5.21	vec2_set CSU.....	721
2.14.3.5.22	vec2_add CSU.....	722
2.14.3.5.23	vec2_sub CSU.....	722
2.14.3.5.24	vec2_dot CSU	722
2.14.3.5.25	vec2_cross CSU.....	722
2.14.3.5.26	vec2_mag CSU	723
2.14.3.5.27	vec2_mag2 CSU	723
2.14.3.5.28	vec2_scale CSU	723
2.14.3.5.29	vec2_copy CSU	724
2.14.3.5.30	vec2_range_squared CSU.....	724
2.14.3.5.31	vec2_norm CSU.....	724
2.14.3.5.32	vec2_rot90 CSU.....	724
2.14.3.5.33	vec2_rot90minus CSU	725
2.14.3.6	geometry.c CSC.....	725
2.14.3.6.1	line_cross_rect CSU	725
2.14.3.6.2	intersect_rect CSU	726
2.14.3.6.3	init_rect CSU	726
2.14.3.6.4	inflate_rect CSU.....	726
2.14.3.6.5	make_polygon CSU	727
2.14.3.6.6	point_in_polygon CSU.....	727
2.14.3.7	random.c CSC.....	728
2.14.3.7.1	get_me_a_random_fraction CSU.....	728

	2.14.3.7.2	check_prob CSU	728
2.14.3.8	cstring.c	CSC.....	729
	2.14.3.8.1	upcase CSU.....	729
	2.14.3.8.2	cistrcmp CSU.....	729
	2.14.3.8.3	cistrncmp CSU	730
2.14.3.9	libutil.h	CSU.....	730
2.14.4	libcomm	CSC.....	731
2.14.4.1	bufpool.c	CSC.....	731
	2.14.4.1.1	simple_queue_enqueue CSU	732
	2.14.4.1.2	simple_queue_dequeue CSU	732
	2.14.4.1.3	simple_queue_length CSU.....	733
	2.14.4.1.4	simple_queue_is_full CSU.....	733
	2.14.4.1.5	simple_queue_is_empty CSU	734
	2.14.4.1.6	simple_queue_is_this_power_of_	
		two CSU	734
	2.14.4.1.7	simple_queue_allocate CSU	734
	2.14.4.1.8	simple_queue_deallocate CSU.....	735
	2.14.4.1.9	simple_queue_describe CSU	735
	2.14.4.1.10	buffer_allocate_from_pool CSU	735
	2.14.4.1.11	buffer_pool_return_buffer CSU.....	736
	2.14.4.1.12	buffer_allocate_into_pool CSU.....	736
	2.14.4.1.13	buffer_pool_allocate CSU.....	737
	2.14.4.1.14	buffer_simple_enqueue CSU	737
	2.14.4.1.15	buffer_simple_dequeue CSU	738
	2.14.4.1.16	buffer_simple_flush CSU.....	738
	2.14.4.1.17	main CSU	738
2.14.4.2	queue_fix.c	CSC.....	739
	2.14.4.2.1	queue_allocate CSU.....	740
	2.14.4.2.2	queue_length CSU	740
	2.14.4.2.3	_queue_hanger_hang_this CSU.....	740
	2.14.4.2.4	_queue_hanger_unhang CSU.....	741
	2.14.4.2.5	queue_enqueue CSU.....	741
	2.14.4.2.6	queue_dequeue CSU.....	741
	2.14.4.2.7	queue_excise CSU	742
	2.14.4.2.8	queue_traverse_and_apply CSU	742
	2.14.4.2.9	queue_flush_hangers CSU.....	742
	2.14.4.2.10	queue_flush CSU	743
	2.14.4.2.11	queue_describe CSU.....	743
	2.14.4.2.12	buffer_allocate CSU	743
	2.14.4.2.13	buffer_set_size CSU.....	744
	2.14.4.2.14	buffer_deallocate CSU	744
	2.14.4.2.15	buffer_deallocate CSU.....	744
	2.14.4.2.16	buffer_length CSU	745

2.14.3.5.2	print_vector CSU.....	716
2.14.3.5.3	make_vector_from_angle_ magnitude CSU.....	716
2.14.3.5.4	angle_clip CSU.....	716
2.14.3.5.5	clip_angle_positive CSU.....	717
2.14.3.5.6	clip_angle_negative CSU.....	717
2.14.3.5.7	angle_between_vectors CSU.....	717
2.14.3.5.8	interior_angle_between_vectors CSU.....	718
2.14.3.5.9	which_side CSU.....	718
2.14.3.5.10	range_squared CSU.....	718
2.14.3.5.11	round_real_to_int CSU.....	719
2.14.3.5.12	vector_z_rotate CSU.....	719
2.14.3.5.13	fvec_to_rvec CSU.....	719
2.14.3.5.14	rvec_to_fvec CSU.....	720
2.14.3.5.15	rmat_to_fmat CSU.....	720
2.14.3.5.16	fmat_to_rmat CSU.....	720
2.14.3.5.17	fvec_copy CSU.....	720
2.14.3.5.18	copy_matrix_row_to_vector CSU....	721
2.14.3.5.19	vec2_print CSU.....	721
2.14.3.5.20	vec2_init CSU.....	721
2.14.3.5.21	vec2_set CSU.....	721
2.14.3.5.22	vec2_add CSU.....	722
2.14.3.5.23	vec2_sub CSU.....	722
2.14.3.5.24	vec2_dot CSU.....	722
2.14.3.5.25	vec2_cross CSU.....	722
2.14.3.5.26	vec2_mag CSU.....	723
2.14.3.5.27	vec2_mag2 CSU.....	723
2.14.3.5.28	vec2_scale CSU.....	723
2.14.3.5.29	vec2_copy CSU.....	724
2.14.3.5.30	vec2_range_squared CSU.....	724
2.14.3.5.31	vec2_norm CSU.....	724
2.14.3.5.32	vec2_rot90 CSU.....	724
2.14.3.5.33	vec2_rot90minus CSU.....	725
2.14.3.6	geometry.c CSC.....	725
2.14.3.6.1	line_cross_rect CSU.....	725
2.14.3.6.2	intersect_rect CSU.....	726
2.14.3.6.3	init_rect CSU.....	726
2.14.3.6.4	inflate_rect CSU.....	726
2.14.3.6.5	make_polygon CSU.....	727
2.14.3.6.6	point_in_polygon CSU.....	727
2.14.3.7	random.c CSC.....	728
2.14.3.7.1	get_me_a_random_fraction CSU.....	728

	2.14.3.7.2	check_prob CSU	728
2.14.3.8	cstring.c CSC.....		729
	2.14.3.8.1	upcase CSU	729
	2.14.3.8.2	cistrcmp CSU	729
	2.14.3.8.3	cistrncmp CSU	730
2.14.3.9	libutil.h CSU		730
2.14.4	libcomm CSC.....		731
2.14.4.1	bufpool.c CSC.....		731
	2.14.4.1.1	simple_queue_enqueue CSU	732
	2.14.4.1.2	simple_queue_dequeue CSU	732
	2.14.4.1.3	simple_queue_length CSU	733
	2.14.4.1.4	simple_queue_is_full CSU	733
	2.14.4.1.5	simple_queue_is_empty CSU	734
	2.14.4.1.6	simple_queue_is_this_power_of_	
		two CSU	734
	2.14.4.1.7	simple_queue_allocate CSU	734
	2.14.4.1.8	simple_queue_deallocate CSU	735
	2.14.4.1.9	simple_queue_describe CSU	735
	2.14.4.1.10	buffer_allocate_from_pool CSU	735
	2.14.4.1.11	buffer_pool_return_buffer CSU	736
	2.14.4.1.12	buffer_allocate_into_pool CSU	736
	2.14.4.1.13	buffer_pool_allocate CSU	737
	2.14.4.1.14	buffer_simple_enqueue CSU	737
	2.14.4.1.15	buffer_simple_dequeue CSU	738
	2.14.4.1.16	buffer_simple_flush CSU	738
	2.14.4.1.17	main CSU	738
2.14.4.2	queue_fix.c CSC.....		739
	2.14.4.2.1	queue_allocate CSU	740
	2.14.4.2.2	queue_length CSU	740
	2.14.4.2.3	_queue_hanger_hang_this CSU	740
	2.14.4.2.4	_queue_hanger_unhang CSU	741
	2.14.4.2.5	queue_enqueue CSU	741
	2.14.4.2.6	queue_dequeue CSU	741
	2.14.4.2.7	queue_excise CSU	742
	2.14.4.2.8	queue_traverse_and_apply CSU	742
	2.14.4.2.9	queue_flush_hangers CSU	742
	2.14.4.2.10	queue_flush CSU	743
	2.14.4.2.11	queue_describe CSU	743
	2.14.4.2.12	buffer_allocate CSU	743
	2.14.4.2.13	buffer_set_size CSU	744
	2.14.4.2.14	buffer_deallocate CSU	744
	2.14.4.2.15	buffer_deallocate CSU	744
	2.14.4.2.16	buffer_allocate CSU	745

2.14.4.2.17	buffer_up_refcnt CSU.....	745
2.14.4.2.18	buffer_enqueue CSU	745
2.14.4.2.19	buffer_excise CSU.....	746
2.14.4.2.20	buffer_dequeue CSU	746
2.14.4.2.21	buffer_traverse_and_apply CSU	746
2.14.4.2.22	buffer_traverse_and_apply_n_	
	times CSU.....	747
2.14.4.2.23	buffer_flush CSU	747
2.14.4.2.24	buffer_describe CSU	747
2.14.4.2.25	queue_distribute CSU	748
2.14.4.2.26	buffer_distribute CSU	748
2.14.4.2.27	buffer_statistics_init CSU.....	748
2.14.4.2.28	buffer_one_more CSU	748
2.14.4.2.29	buffer_one_less CSU.....	748
2.14.4.2.30	buffer_statistics_print CSU	749
2.14.4.3	libcomm.h CSU	749
2.15	SUPPORT CSC.....	750
2.15.1	blaster.c CSC.....	750
2.15.1.1	main CSU	751
2.15.1.2	build_packets CSU.....	751
2.15.1.3	do_send CSU.....	752
APPENDIX A.....		A-1
APPENDIX B.....		B-1
APPENDIX C.....		C-1
APPENDIX D.....		D-1
APPENDIX E.....		E-1
INDEX BY SECTION NUMBER		Index-1

1. INTRODUCTION : SAF SIMULATION HOST CSCI

1.1 BACKGROUND

The Semi-automated Forces (SAF) system provides the means of incorporating unmanned simulations of vehicles into a large-scale battle training simulation; therefore, it allows for a realistic representation of enemy forces, as well as portions of friendly forces, without requiring human crews to operate each component of these forces, thus eliminating the need for excessive numbers of personnel. These intelligent, computer-operated participants, called SAF vehicles, behave realistically enough that observers on the network are unable to distinguish them from manned simulations. The SAF system provides command and control for up to a battalion of Semi-automated vehicles.

Since the detailed simulation of many realistic vehicles is a computationally intensive task, the SAF system is typically partitioned across several processing platforms. One platform provides the user interface and another provides the vehicle simulation. This division is reflected in the software interface organization depicted in Figure 1.2-1.

The SAF Simulation Host (Simhost) CSCI (Computer Software Configuration Item) simulates the SAF vehicles and units. It receives commands from one or more workstations for the initialization, command, and control of SAF units. It simulates the vehicles and units interacting with each other and the environment, and executes their received commands. In addition, the SAF Simhost CSCI monitors the activities of remote vehicles simulated by other computers connected to the SIMNET Local Area Network (LAN), and simulates the interactions between these remote vehicles and the locally simulated vehicles. It also sends packets out on the SIMNET LAN describing the state of the locally simulated vehicles and the interactions with the remote vehicles.

1.2 EXTERNAL INTERFACES

Figure 1.2-1 depicts the software interface organization of the SAF Sim Host CSCI [8.0] with the SAF Workstation CSCI [6.0] and the SAF Parameter Editor CSCI [7.0].

The SAF Simhost CSCI provides an interface between the SAF Segment and the SIMNET System via the SIMNET 6.6 Protocol. This protocol is not significantly different from the SIMNET 6.0 Protocol documented in BBN Report No. 7102, titled "The SIMNET Network and Protocols."

The SAF Simhost CSCI [8.0] interfaces with the SAF Workstation CSCI [6.0] via the SAF Command Protocol. This protocol is described in the SAF Command Protocol Appendix, Appendix A in the SAF Workstation CSCI.

The SAF Simhost CSCI [8.0] interfaces with the SAF Parameter Editor CSCI [7.0] via the SAF parameter files. These files are described in the SAF Parameter Files Appendix, Appendix B in the SAF Workstation CSCI.

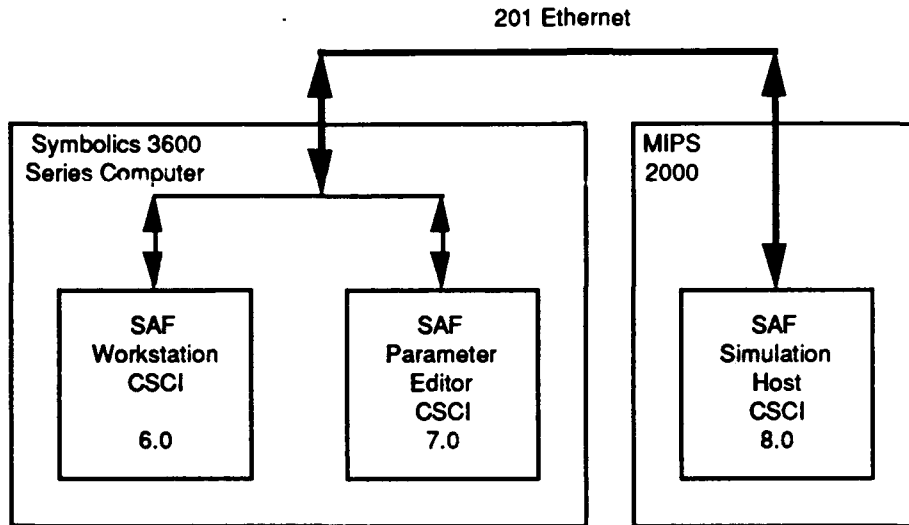


Figure 1.2-1: Software Interface Organization

In addition, the SAF Simhost CSCI shares a terrain database with the other SIMNET Segments. The terrain representation used by the SAF Simhost CSCI is documented in the SAF Terrain Files Appendix.

1.3 INTERNAL STRUCTURE

Chronologically, the development of the SAF system post-dates the development of the vehicle simulation software. Several of the CSCs that compose the SAF Simhost CSCI are shared with other SIMNET systems.

The structure of the SAF Simhost CSCI [8.0] is shown in Figure 1.3-1. There are 15 top-level CSCs.

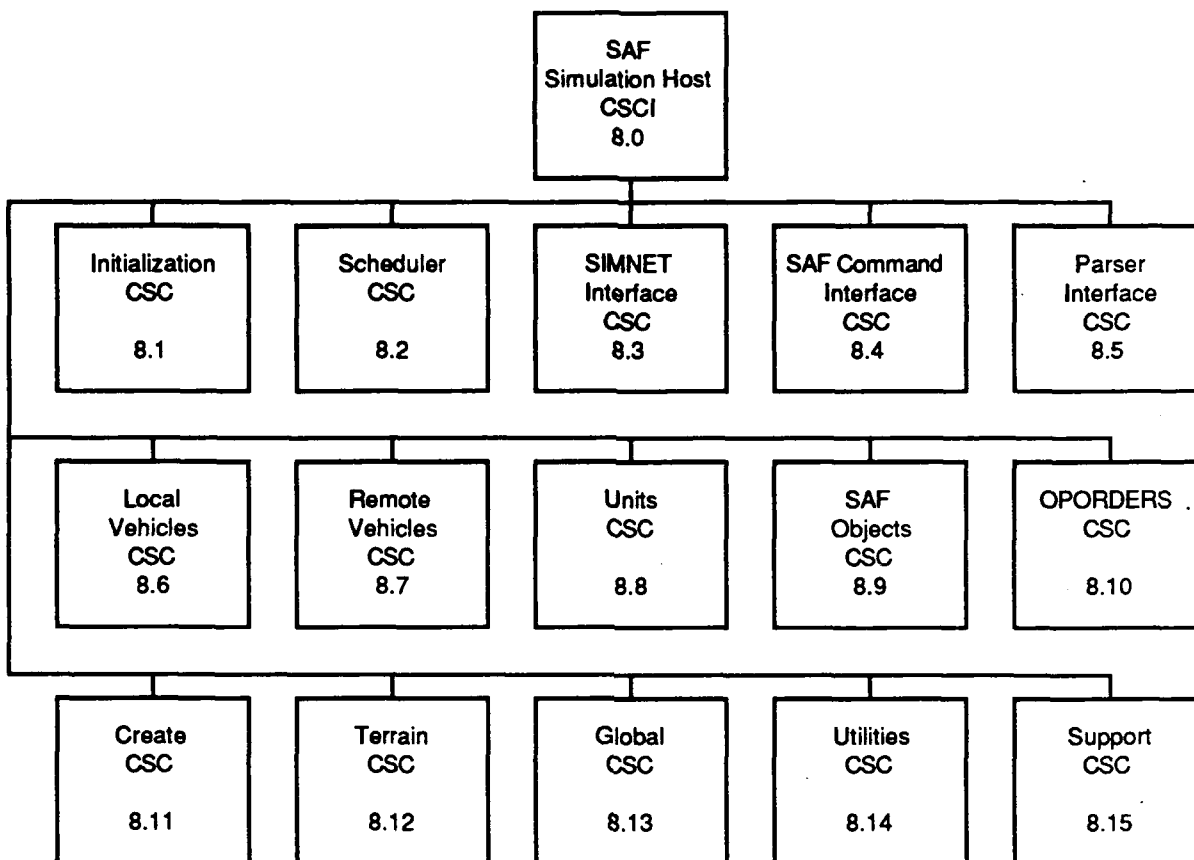


Figure 1.3-1: SAF Simhost CSCI Structure

1.4 CONFIGURATION AND CONFIGURATION MANAGEMENT

The SAF Simhost CSCI is written in the C programming language and executes on the SAF Simhost HWCI, currently a MIPS 2000 computer. It executes under the MIPS RISC/os operating system (a UNIX variant based on AT&T System 5 with Berkeley extensions). The code for the SAF Simhost CSCI has been written to maximize portability across UNIX computers. Variants of this code minus the SIMNET LAN interface have been run on SUN and Silicon Graphics computers. Portions of the SIMNET interface code run on an intelligent network controller (CMC ENP100 VME card, with 68020 processor).

1.5 TERMINOLOGY AND DOCUMENTATION

CIS	Combat Instruction Set
CM	Control Measure
CSC	Computer Software Component
CSCI	Computer Software Configuration Item
CSU	Computer Software Unit
LAN	Local Area Network
IVIS	Inter-vehicular Interface System
MCC	Measurement, Command and Control
OPORDERS	Operations Orders
PAE	Position, Appearance and Echelon
RUDP	Reliable UDP
SAF	Semi-automated Forces
sbx	Symbolics (Computer)
Simhost	Simulator Host
SIMNET	Simulation Network
UDP	User Datagram Protocol
VAP	Vehicle Appearance Packets

2. CSC DESCRIPTIONS

The SAF Simulation Host CSCI [8.0] contains 15 top level CSCs. They are listed in Figure 1.3-1.

2.1. INITIALIZATION CSC

The initialization code starts up the Simhost CSCI. It processes the command line switches, initializes global variables, loads the parameter and terrain files, connects to the SIMNET interface controller, opens sockets for communications with SAF Workstation CSCIs and then starts up the scheduler. The scheduler does not return to the operating system. Exit only occurs via the SIGINT signal (exit(0)).

The top level Initialization CSC has three secondary level CSCs, as shown in Figure 2.1-1: the library libreader [8.1.1], which reads the parameter files; the Parameters CSC [8.1.2], which handles the symbol definition and storage; and the Initialization CSC [8.1.3], which contains the global variable definitions, startup programs, and version information.

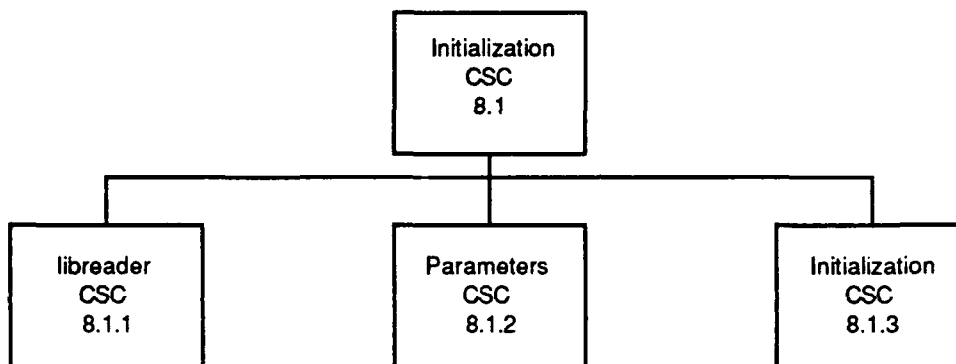


Figure 2.1-1: Initialization CSC Structure

2.1.1 libreader CSC

/simnet/libsrc/libreader

This library handles the reading of the parameter files for use by the Simhost program. Figure 2.1-2 depicts the five CSCs and CSUs contained in the libreader CSC.

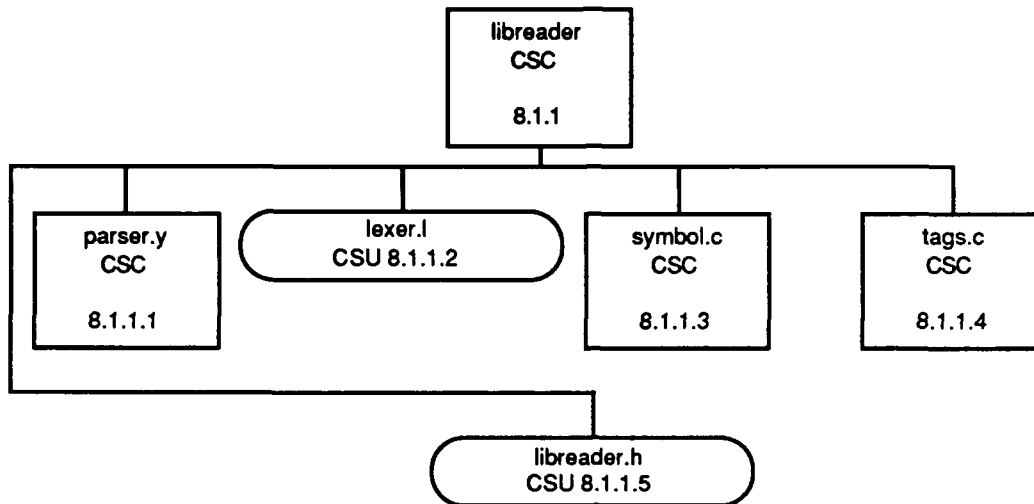


Figure 2.1-2: libreader CSC Structure

2.1.1.1 parser.y CSC

/simnet/libsrc/libreader/parser.y

This CSC contains the yacc sources to build the parser. It contains seven CSUs, shown in Figure 2.1-3 depicting the structure of the parser.c CSC.

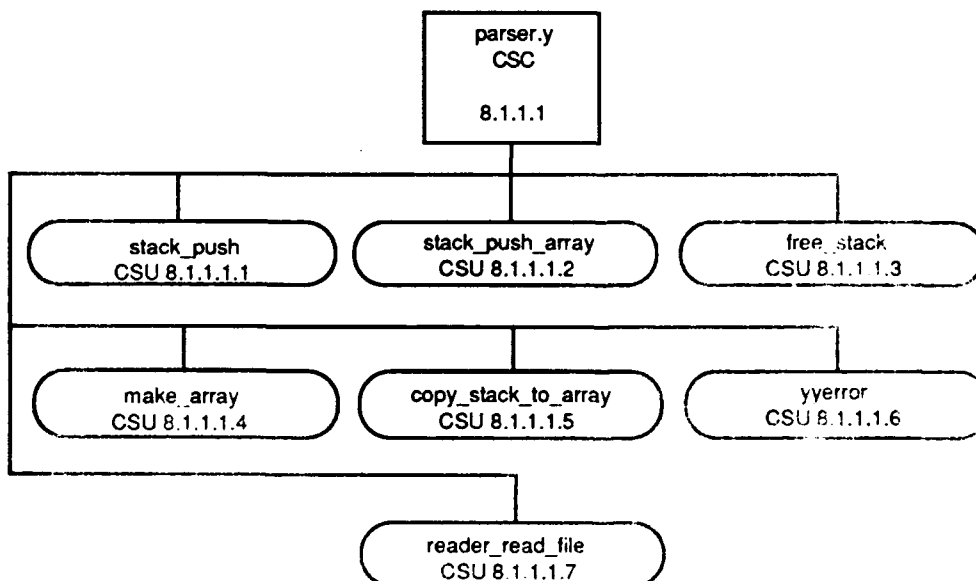


Figure 2.1-3: parser.y CSC Structure

2.1.1.1.1 stack_push CSU

The CSU allocates a new cell, sets its data value, and pushes it onto the passed stack.

Parameters		
Parameters	Type	Where Typedef Declared
data_ptr	pointer to a DATA UNION	Sec. 2.1.1.5
stack_ptr	pointer to a READER_STACK	Sec. 2.1.1.5
ReturnValues		
Return Value	Type	Meaning
new_stack	pointer to READER_STACK	Pointer to last stack built
Calls		
Function	Where Described	
ALLOCATOR	Sec. 2.1.1.5 See Appendix A	

Table 2.1-1: stack_push CSU [8.1.1.1.1]

2.1.1.1.2 stack_push_array CSU

This CSU pushes the array of DATA_UNIONs onto the READER_STACKs.

Parameters		
Parameters	Type	Where Typedef Declared
arr	pointer to a DATA UNION	Sec. 2.1.1.5
stack_ptr	pointer to a READER_STACK	Sec. 2.1.1.5
ReturnValues		
Return Value	Type	Meaning
stack_ptr	READER_STACK	The last stack pointer
Calls		
Function	Where Described	
stack_push	Sec. 2.1.1.1.1	

Table 2.1-2: stack_push_array CSU [8.1.1.1.2]

2.1.1.1.3 free_stack CSU

This CSU frees READER_STACK memory provided there are more stacks (pointers) to free.

Parameters		
Parameters	Type	Where Typedef Declared
stack_pointer	pointer to READER_STACK	Sec. 2.1.1.5

Calls	
Function	Where Described
free_stack	Sec. 2.1.1.1.3
DEALLOCATE	Sec. 2.1.1.5 See Appendix A

Table 2.1-3: free_stack CSU [8.1.1.1.3]

2.1.1.1.4 make_array CSU

The CSU determines the number of DATA_UNION structures required and requests enough memory to make this array. It then transfers the READER_STACK pointer to the DATA_UNION pointers via a call to copy_stack_to_array.

Parameters		
Parameters	Type	Where Typedef Declared
stack_pointer	pointer to READER_STACK	Sec. 2.1.1.5
ReturnValues		
Return Value	Type	Meaning
result	pointer to DATA_UNION	Where array is finished
Calls		
Function	Where Described	
ALLOCATOR	Sec. 2.1.1.5 See Appendix A	
copy_stack_to_array	Sec. 2.1.1.1.5	

Table 2.1-4: make_array CSU [8.1.1.1.4]

2.1.1.1.5 copy_stack_to_array CSU

This CSU copies pointers from the READER_STACKs to the DATA_UNIONs until it exhausts the number of READER_STACKs.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to READER_STACK	Sec. 2.1.1.5
r	pointer to DATA_UNION	Sec. 2.1.1.5
ReturnValues		
Return Value	Type	Meaning
next_pos+1	pointer to DATA_UNION	Next point to stuff
r	pointer to DATA_UNION	Out of reader stack
Calls		
Function	Where Described	
copy_stack_to_array	Sec. 2.1.1.1.5	
DEALLOCATOR	Sec. 2.1.1.5 See Appendix A	

Table 2.1-5: copy_stack_to_array CSU [8.1.1.1.5]

2.1.1.1.6 yyerror CSU

This CSU prints a parsing error message including file name, line number, and error. The parameter passed is a pointer to the error message.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to char	Standard

Table 2.1-6: yyerror CSU [8.1.1.1.6]

2.1.1.1.7 reader_read_file CSU

The CSU reader_read_file invokes the parser on the file named fname, and puts the results of parsing into the DATA_UNION pointed to by du.

Parameters		
Parameters	Type	Where Typedef Declared
fname	pointer to char	Standard
du	pointer to Data Union	Sec. 2.1.1.5
ReturnValues		
Return Value	Type	Meaning
1	int	Successful
0	int	Error (see below)
Errors		
Error Name	Reason for Error	
0 (Return value)	Can't open file, or error condition occurred during parsing	

Table 2.1-7: reader_read_file CSU [8.1.1.1.7]

2.1.1.2 lexer.l CSU

/simnet/libsrc/libreader/lexer.l

This CSU contains the lex sources to build the lexer, including both the definitions for the symbols read and the actions taken based on the definition encountered.

2.1.1.3 symbol.c CSC

/simnet/libsrc/libreader/symbol.c

This CSC contains the symbol table utility. Which version, if any, of each CSU compiled within this file depends on whether NO_SYMBOLS is defined as an option of the Makefile.

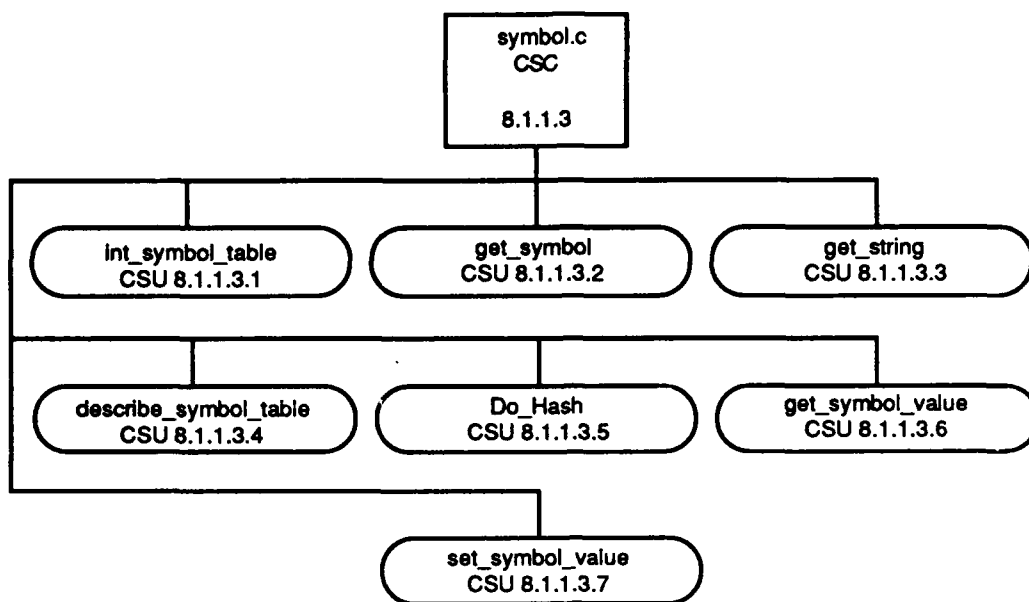


Figure 2.1-4: symbol.c CSC Structure

In addition to the seven CSUs, symbol.c contains a structure tagged bucket_entry and three constant definitions required by the CSU Do_Hash, Sec. 2.1.1.3.5. (See the following two tables.) These are only compiled on the condition that NO_SYMBOLS is undefined as an option of the Makefile, which mean that symbols are used.

Item	Type	Where Type Defined
symbol	pointer to char	Standard
next	pointer to struct BUCKET_ENTRY	This structure tag

Table 2.1-8: BUCKET_ENTRY Structure Definition

The BUCKET_ENTRY structure contains a pointer to the current string and a pointer to the next entry. Following the structure is the definition that the pointer to the symbol table array (symbol_table[SYMBOL_TABLE_SIZE]) is a static BUCKET_ENTRY.

Constant	Value
MAX_HASH_CHARS	5
LENGTH_MULTIPLIER	5
SHIFT_AMOUNT	2

Table 2.1-9: symbol.c Constant Definitions

2.1.1.3.1 init_symbol_table CSU

This CSU initializes the symbol table. It is only compiled on the condition that NO_SYMBOLS is undefined as an option of the Makefile.

2.1.1.3.2 get_symbol CSU

This CSU finds the symbol in the symbol table that matches the string "s". It makes a new entry if no is found. It returns a pointer to the string.

The CSU Do_Hash is called only in the version of get_symbol that is compiled on the condition that NO_SYMBOLS is undefined as an option of the Makefile.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to char	Standard
ReturnValues		
Return Value	Type	Meaning
bucket->symbol	pointer to char	If NO_SYMBOLS undefined
copy	pointer to char	If NO_SYMBOLS defined
Calls		
Function	Where Described	
Do Hash	Sec. 2.1.1.3.5	
ALLOCATOR	Sec. 2.1.1.5 See Appendix A	

Table 2.1-10: get_symbol CSU [8.1.1.3.2]

2.1.1.3.3 get_string CSU

The CSU get_symbol is called only in the version of get_string that is compiled on the condition that NO_SYMBOLS is undefined as an option of the Makefile; otherwise, the version of get_string that uses the macro ALLOCATOR is used. In either condition (with or without symbols), the CSU gets a copy of the string pointed to by the CSU parameter and returns a pointer to the copy of the string.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to char	Standard
ReturnValues		
Return Value	Type	Meaning
get_symbol(s no_quotes+1)	pointer to char	pointer to copy of string
copy	pointer to char	pointer to copy of string
Calls		
Function	Where Described	
get_symbol	Sec. 2.1.1.3.2	
ALLOCATOR	Sec. 2.1.1.5 See Appendix A	

Table 2.1-11: get_string CSU [8.1.1.3.3]

2.1.1.3.4 describe_symbol_table CSU

This CSU is compiled on the condition that NO_SYMBOLS is undefined as an option of the Makefile. This CSU calculates and prints a hash table summary, returning the number of entries per bucket. It also prints a symbol table if the input parameter "printp" is true.

Parameters		
Parameters	Type	Where Typedef Declared
printp	int	Standard
ReturnValues		
Return Value	Type	Meaning
sym_count/bucket_count	double	entries per bucket

Table 2.1-12: describe_symbol_table CSU [8.1.1.3.4]

2.1.1.3.5 Do_Hash CSU

This CSU is compiled on the condition that NO_SYMBOLS is undefined as an option of the Makefile. It generates a hash index and places it into the symbol table based on the string passed.

Parameters		
Parameters	Type	Where Typedef Declared
String_Ptr	pointer to char	Standard
String_length	int	Standard
ReturnValues		
Return Value	Type	Meaning
Value	int	hash index into the symbol table

Table 2.1-13: Do_Hash CSU [8.1.1.3.5]

2.1.1.3.6 get_symbol_value CSU

If NO_SYMBOLS is defined, this CSU returns a NULL pointer; otherwise, it returns the macro value associated with the symbol.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to char	Standard
ReturnValues		
Return Value	Type	Meaning
NULL	pointer to DATA_UNION	NO_SYMBOLS defined
((DATA_UNION **) (s-(sizeof(DATA_UNION*))))	pointer to DATA_UNION	NO_SYMBOLS undefined, pointer to symbol value string

Table 2.1-14: get_symbol_value CSU [8.1.1.3.6]

2.1.1.3.7 set_symbol_value CSU

This CSU associates a macro value with a symbol name.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to char	Standard
v	pointer to DATA UNION	Sec. 2.1.1.5

Table 2.1-15: set_symbol_value CSU [8.1.1.3.7]

2.1.1.4 tags.c CSC

/simnet/libsrc/libreader/tags.c

This CSC contains the tagged array lookup CSUs. The structure of the tags.c CSC, containing seven CSUs, is depicted in Figure 2.1-5.

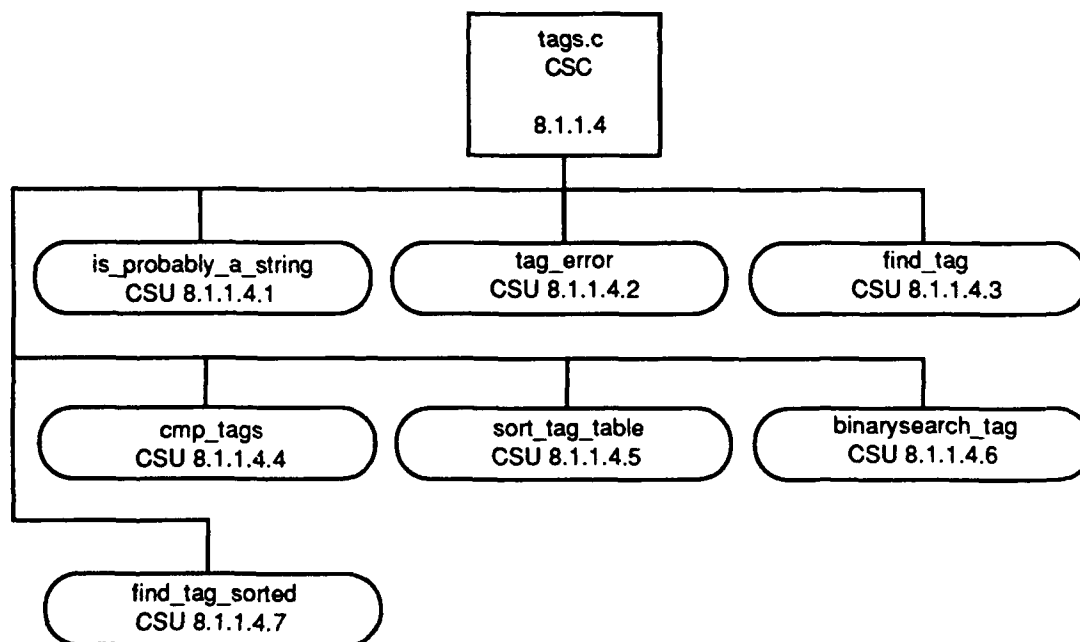


Figure 2.1-5: tags.c CSC Structure

2.1.1.4.1 is_probably_a_string CSU

This CSU is used in error message generation to determine if a pointer likely points to an ASCII string.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to char	Standard

ReturnValues		
Return Value	Type	Meaning
0	static int	
(s == 0)	static int	

Table 2.1-16: `is_probably_a_string` CSU [8.1.1.4.1]2.1.1.4.2 `tag_error` CSU

This CSU reports that a tag was not found in the passed table.

Parameters		
Parameters	Type	Where Typedef Declared
tag	pointer to char	Standard
table	pointer to DATA_UNION	Sec. 2.1.1.5
tagged	int	Standard
errlevel	int	Standard
Calls		
Function	Where Described	
<code>is_probably_a_string</code>	Sec. 2.1.1.4.1	

Table 2.1-17: `tag_error` CSU [8.1.1.4.2]2.1.1.4.3 `find_tag` CSU

This CSU uses a linear search through the table, a DATA_UNION array, (skipping the first entry if this entry is tagged TABLE_TAGGED) until it finds the entry that has "tag" as its first element.

There are three error levels possible, as passed by the error_level parameter. In all cases the return is zero. Their descriptions are presented in Table 2.1-19.

Error Level	Meaning
TAGS_NO_ERRORS	Do not report errors
TAGS_REPORT_ERRORS	Report "tag not found" errors
TAGS_ERRORS_WITH_CONTEXT	Report "tag not found" errors and give examples of the tags which were compared against

Table 2.1-18: `find_tag` Error Levels

Parameters		
Parameters	Type	Where Typedef Declared
tag	pointer to char	Standard
table	pointer to DATA_UNION	Sec. 2.1.1.5
tagged	int	Standard
error_level	int	Standard

ReturnValues		
Return Value	Type	Meaning
entry	pointer to DATA_UNION	The entry where tag was found
0	pointer to DATA_UNION	Tag not found
Calls		
Function	Where Described	
symbols_match	Sec. 2.1.1.5	
tag_error	Sec. 2.1.1.4.2	

Table 2.1-19: find_tag CSU [8.1.1.4.3]

2.1.1.4.4 cmp_tags CSU

This CSU returns a value greater than, equal to, or less than zero depending upon whether the first passed symbol is considered greater than, equal to, or less than the second in sorting.

Parameters		
Parameters	Type	Where Typedef Declared
du0	pointer to DATA_UNION	Sec. 2.1.1.5
du1	pointer to DATA_UNION	Sec. 2.1.1.5
ReturnValues		
Return Value	Type	Meaning
-1	static int	First passed symbol is less than the second in the string
1	static int	First passed symbol is greater than the second in the string
0	static int	First passed symbol is equal to the second in the string
Calls		
Function	Where Described	
symbol_compare	Sec. 2.	

Table 2.1-20: cmp_tags CSU [8.1.1.4.4]

2.1.1.4.5 sort_tag_table CSU

This CSU uses qsort to sort the specified "table" (skipping the first entry if this entry "tagged" is tag itself), keyed by the "tag" of each entry.

Parameters		
Parameters	Type	Where Typedef Declared
table	pointer to DATA_UNION	Sec. 2.1.1.5
tagged	int	Standard

Calls	
Function	Where Described
qsort	Sec. 2.

Table 2.1-21: sort_tag_table CSU [8.1.1.4.5]

2.1.1.4.6 binarysearch_tag CSU

This CSU conducts a recursive binary search for the tag in the passed table.

Parameters		
Parameters	Type	Where Typedef Declared
key	pointer to char	Standard
first	pointer to DATA UNION	Sec. 2.1.1.5
last	pointer to DATA UNION	Sec. 2.1.1.5
ReturnValues		
Return Value	Type	Meaning
NULL	static pointer to DATA UNION	
first	static pointer to DATA UNION	
last	static pointer to DATA UNION	
binarysearch_tag	static pointer to DATA UNION	
Calls		
Function	Where Described	
symbols_compare	Sec. 2.1.1.5	
binarysearch_tag	Sec. 2.1.1.4.6	

Table 2.1-22: binarysearch_tag CSU [8.1.1.4.6]

2.1.1.4.7 find_tag_sorted CSU

This CSU searches through the table (skipping the first entry if this entry "tagged" is table tagged) until it finds the entry that has "tag" as its first element. It is similar to find_tag except that the table is presumed to be sorted. Thus, it uses a binary search (binarysearch_tag) rather than a linear one.

Parameters		
Parameters	Type	Where Typedef Declared
tag	pointer to char	Standard
table	pointer to DATA UNION	Sec. 2.1.1.5
tagged	int	Standard
errlevel	int	Standard
ReturnValues		
Return Value	Type	Meaning
result -> array	pointer to DATA UNION	
0	pointer to DATA UNION	tag error

Calls	
Function	Where Described
binarysearch_tag	Sec. 2.1.1.4.6
tag_error	Sec. 2.1.1.4.2

Table 2.1-23: find_tag_sorted CSU [8.1.1.4.7]

2.1.1.5 libreader.h CSU

/simnet/libsrc/libreader/libreader.h

This header file CSU contains external CSU defines, a typedef union, a typedef struct, five tag constants, a symbol table size constant, and conditional macro defines. The macros are in Appendix A.

The following typedef union is tagged data_union.

Item	Type	Where Type Defined
charptr	pointer to char	Standard
integer	int	Standard
real	float	Standard
array	pointer to a union data_union	This union structure

Table 2.1-24: DATA_UNION Union Definition

The following typedef struct, used only for parsing, is tagged stack. Note that "next" points to the next stack.

Item	Type	Where Type Defined
data	DATA_UNION	Union above this definition
next	pointer to struct stack	This structure

Table 2.1-25: READER_STACK Structure Definition

The following table contains the TAGS and TABLE_TAGGED constants.

Constant	Value
TAGS NO ERRORS	0
TAGS REPORT ERRORS	1
TAGS ERRORS WITH CONTEXT	2
TABLES NOT_TAGGED	0
TABLES TAGGED	1

Table 2.1-26: TAGS and TABLE_TAGGED Constant Definitions

The following symbol table size constant is predicated on the size not having been previously defined. This would occur as part of the Makefile for the library.

Constant	Value
SYMBOL_TABLE_SIZE	947

Table 2.1-27: SYMBOL_TABLE_SIZE Constant Definition

2.1.2 Parameters CSC

This CSC defines and initializes the Simhost code symbols (list of parameters). It consists of the symbols.c CSC and symbols.h CSU, as seen in the structure depicted in Figure 2.1-6.

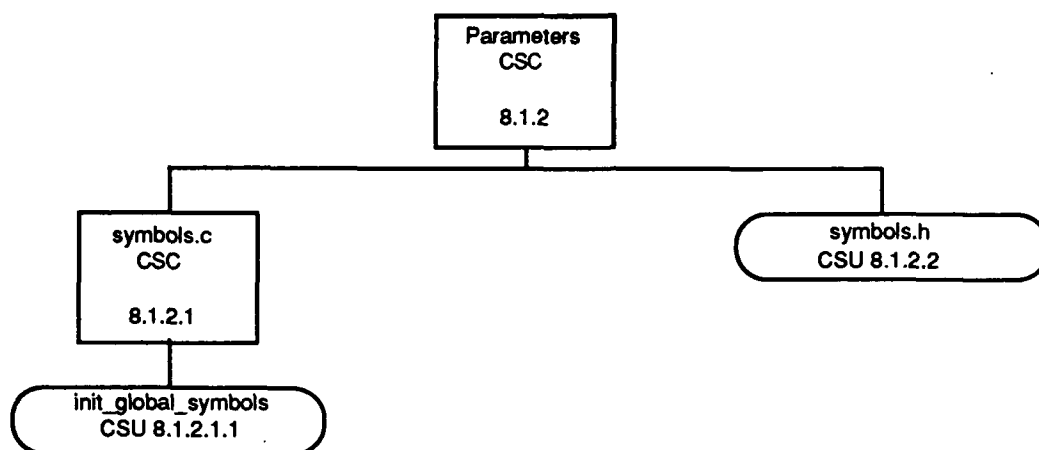


Figure 2.1-6: Parameters CSC Structure

2.1.2.1 symbols.c CSC

/simnet/src/host/symbols.c

The file contains a single CSU, `init_global_symbols`. It acts as the storage location for the symbols (common strings stored in a shared area) used in the Simhost code, and also handles initializing the symbols.

2.1.2.1.1 init_global_symbols CSU

This CSU initializes the `*_SYM` symbols to their character string values for the other files composing the Simhost code.

Calls	
Function	Where Described
get_symbol	Sec. 2.1.1.3.2

Table 2.1-28: init_global_symbols CSU [8.1.2.1.1]

2.1.2.2 symbols.h CSU

/simnet/src/host/symbols.h

This CSU defines the symbols for the other files composing the Simhost code.

2.1.3 Initialization CSC

This CSC provides the initialization of the Simhost program. It consists of the main.c CSC and the version.c CSC. The structure of the Initialization CSC is depicted in Figure 2.1-7.

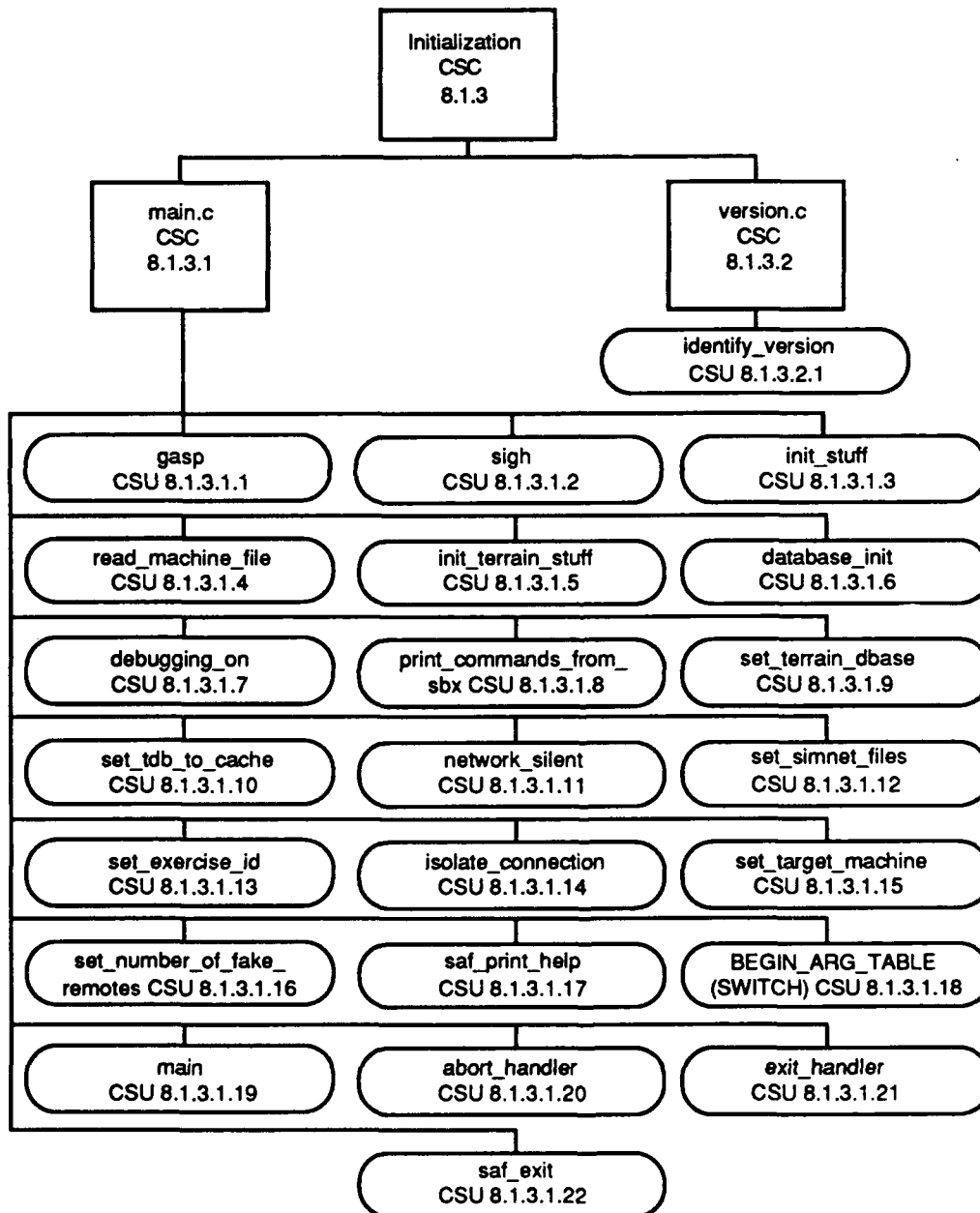


Figure 2.1-7: Initialization CSC Structure

2.1.3.1 main.c CSC

/simnet/src/host/main.c

This CSC contains the definitions for the global variables used throughout the program. It also contains the code to initialize the various systems and data structures used during the execution of the program. This initialization includes loading data files, creating tables, starting clocks and the scheduler, and initializing the terrain database. Other items included in this file are CSUs to monitor the performance level of the program and handle overload situations, the code to handle arguments at program start up time, the program abort handler, and of course, the CSU main.

2.1.3.1.1 gasp CSU

Calls	
Function	Where Described
buffer allocate	Sec. 2.14.4.2.12
fill sbx opfor header	Sec. 2.4.3.2.18
write buffer all sbx	Sec. 2.4.3.2.3
buffer deallocate	Sec. 2.14.4.2.15

Table 2.1-29: gasp CSU [8.1.3.1.1]

2.1.3.1.2 sigh CSU

The sigh CSU prints a "Sigh. All better" message to the screen.

2.1.3.1.3 init_stuff CSU

This CSU calls other initialization routines in the proper order.

ReturnValues		
Return Value	Type	Meaning
SUCCESS	int	initialization succeeded
Calls		
Function	Where Described	
heap_create	Sec. 2.14.2.2.2	
exit_handler	Sec. 2.1.3.1.21	
init clocks	Sec. 2.14.3.3.1	
scheduler init	Sec. 2.2.1.3.1	
set_critical_performance_level	Sec. 2.2.1.4.3	
init symbol table	Sec. 2.1.1.3.1	
init safobj table	Sec. 2.2.2.1	
init saf to simnet id table		
FORCE_OUT	Sec. 2.5.2.2	
init global symbols	Sec. 2.1.2.2	
database init	Sec. 2.1.3.1.6	
init terrain stuff	Sec. 2.1.3.1.5	
read machine file	Sec. 2.1.3.1.4	

Table 2.1-30: init_stuff CSU [8.1.3.1.3]

2.1.3.1.4 read_machine_file CSU

This CSU reads machine-specific information and stores it in global variables.

Calls	
Function	Where Described
reader_read file	Sec. 2.1.1.1.7
ERROR OUT	Sec. 2.5.2.2
find tag	Sec. 2.1.1.4.3
get symbol	Sec. 2.1.1.3.2
open io connections	Sec. 2.4.3.1
FORCE OUT	Sec. 2.5.2.2

Table 2.1-31: read_machine_file CSU [8.1.3.1.4]

2.1.3.1.5 init_terrain_stuff CSU

This CSU calls terrain-related initialization routines.

Errors	
Error Name	Reason for Error
ERROR OUT	Unable to initiate terrain
Calls	
Function	Where Described
FORCE OUT	Sec. 2.5.2.2
read quadtree database	Sec. 2.12.1.1.1
tdb_init_cache	Sec. 2.21.7.27.1 in MCC CSCI SDD
tdb_error	Sec. 2.21.7.17.1 in MCC CSCI SDD
ERROR OUT	Sec. 2.5.2.2
tdb_init_memory	Sec. 2.21.7.25.1 in MCC CSCI SDD
tdb_init_patch guards	Sec. 2.21.7.10.4 in MCC CSCI SDD
tdb_get_db_name	Sec. 2.21.7.15.19 in MCC CSCI SDD
init_grid_tables	Sec. 2.9.3.1.10

Table 2.1-32: init_terrain_stuff CSU [8.1.3.1.5]

2.1.3.1.6 database_init CSU

This CSU initializes global variables.

Calls	
Function	Where Described
reader_read file	Sec. 2.1.1.1.7
database_read	Sec. 2.6.8.8.1
sort_form_db	Sec. 2.8.1.4.12
init_mappings	Sec. 2.4.3.1.1

Table 2.1-33: database_init CSU [8.1.3.1.6]

2.1.3.1.7 debugging_on CSU

This CSU sets all debugging flags on, except event debugging, and prints "Debugging is now on."

2.1.3.1.8 print_commands_from_sbx CSU

This CSU sets g_print_commands to TRUE and prints "Printing command from SBX."

2.1.3.1.9 set_terrain_dbase CSU

This CSU sets the terrain database to load.

Parameters		
Parameters	Type	Where Typedef Declared
dbase	pointer to char	Standard

Table 2.1-34: set_terrain_dbase CSU [8.1.3.1.9]

2.1.3.1.10 set_tdb_to_cache CSU

This CSU enables caching (versus loading the entire object into memory).

2.1.3.1.11 network_silent CSU

This CSU disables use of the SIMNET LAN input/output.

2.1.3.1.12 set_simnet_files CSU

This CSU is included at compile time if FILE_NET is defined. This function is used when SIMNET traffic is simulated via the NFS interface.

Parameters		
Parameters	Type	Where Typedef Declared
filein	pointer to char	Standard
fileout	pointer to char	Standard

Table 2.1-35: set_simnet_files CSU [8.1.3.1.12]

2.1.3.1.13 set_exercise_id CSU

This CSU sets exercise to a number. The default is 1.

Parameters		
Parameters	Type	Where Typedef Declared
strnum	pointer to char	Standard
Calls		
Function	Where Described	
ERROR OUT	Sec. 2.5.2.2	

Table 2.1-36: set_exercise_id CSU [8.1.3.1.13]

2.1.3.1.14 isolate_connection CSU

This CSU specifies that only one port should be opened for input.

Parameters		
Parameters	Type	Where Typedef Declared
strnum	pointer to char	Standard
Calls		
Function	Where Described	
ERROR_OUT	Sec. 2.5.3.3	

Table 2.1-37: isolate_connection CSU [8.1.3.1.14]

2.1.3.1.15 set_target_machine CSU

This CSU, included at compilation time if BENCHMARK is defined, is used only for benchmarking.

Parameters		
Parameters	Type	Where Typedef Declared
strnum	pointer to char	Standard

Table 2.1-38: set_target_machine CSU [8.1.3.1.15]

2.1.3.1.16 set_number_of_fake_remotes CSU

This CSU, included at compilation time if BENCHMARK is defined, is used only for benchmarking.

2.1.3.1.17 saf_print_help CSU

This CSU prints the help menu.

2.1.3.1.18 BEGIN_ARG_TABLE(SWITCH) CSU

This CSU consists of a series of SWITCH macros, defining the actions resulting from the activation of a switch following the printing of the saf_print_help CSU message

Calls	
Function	Where Described
SWITCH	Sec. 2.1.3.1.18
BREAK SET	

Table 2.1-39: BEGIN_ARG_TABLE(SWITCH) CSU [8.1.3.1.18]

2.1.3.1.19 main CSU

This CSU calls the initialization routine and starts the scheduler.

Parameters		
Parameters	Type	Where Typedef Declared
argc	int	Standard
argv[]	pointer to char	Standard
ReturnValues		
Return Value	Type	Meaning
0	int	completed
Calls		
Function	Where Described	
abort handler	Sec. 2.1.3.1.20	
identify_version	Sec. 2.1.3.2.1	
proc switches	Sec. 2.14.3.1.1	
ERROR OUT	Sec. 2.5.2.2	
init stuff	Sec. 2.1.3.1.3	
exit handler	Sec. 2.1.3.1.21	
start simnet	Sec. 2.3.2.1	
parser create	Sec. 2.5.2.1.3	
invoke functions until	Sec. 2.2.1.5.1	

Table 2.1-40: main CSU [8.1.3.1.19]

2.1.3.1.20 abort_handler CSU

This CSU handles the abort signal, cleans up the system functions, and exits.

Parameters		
Parameters	Type	Where Typedef Declared
code	int	Standard
Calls		
Function	Where Described	
getpid		
simnet exit	Sec. 2.3.2.7	
exit all sbx conns	Sec. 2.4.3.2.2	
parser restore term	Sec. 2.5.2.1.2	
heap destroy	Sec. 2.14.2.2.3	

Table 2.1-41: abort_handler CSU [8.1.3.1.20]

2.1.3.1.21 exit_handler CSU

This CSU handles the exit signal, cleans up the system functions, and exits.

Parameters		
Parameters	Type	Where Typedef Declared
code	int	Standard
Calls		
Function	Where Described	
simnet_exit	Sec. 2.3.2.7	
exit_all_sbx_conns	Sec. 2.4.3.2.2	
cache_and_file_terminate	Sec 2.21.7.7.2 in MCC CSCI SDD	
parser_restore_term	Sec. 2.5.2.1.2	
heap_destroy	Sec. 2.14.2.2.3	

Table 2.1-42: exit_handler CSU [8.1.3.1.21]

2.1.3.1.22 saf_exit CSU

This CSU calls two other CSUs, saf_complete_reset and exit_handler, which do the work of exiting.

Calls	
Function	Where Described
saf_complete_reset	Sec. 2.2.2.3.1
exit_handler	Sec. 2.1.3.1.20

Table 2.1-43: saf_exit CSU [8.1.3.1.22]

2.1.3.2 version.c CSC

/simnet/src/host/version.c

This CSC prints the version information when the Simhost CSCI (phantom program) is started. Comments provide a brief history of releases.

2.1.3.2.1 identify_version CSU

Through multiple calls to printf this CSU displays the current version (PHANTOM 3.9.10) and date (Released on 8/30/90).

2.2 SCHEDULER CSC

The Scheduler CSC runs the simulation. To avoid the overhead of system calls and to make the program more portable, the UNIX scheduler is not used. The scheduler "ticks" the Parser Interface CSC, the SIMNET Interface CSC, the SAF Command Interface CSC, the Local Vehicles CSC, the Remote Vehicles CSC, and the Units CSC, thus running the entire simulation and the external interfaces.

Each "tick" is an elapse of time. The time between ticks, or scheduled updates for a function, depends on the periodic rate or next time of operation for that function. The scheduler compares the next time of function operation with the present time, and if the present time is past the scheduled time, the CSU code is executed.

The top level Scheduler CSC consists of four secondary level CSCs: libsched CSC, safobj.c CSC, tickable.c CSC, and saf.c CSC. The Scheduler CSC structure is depicted in Figure 2.2-1.

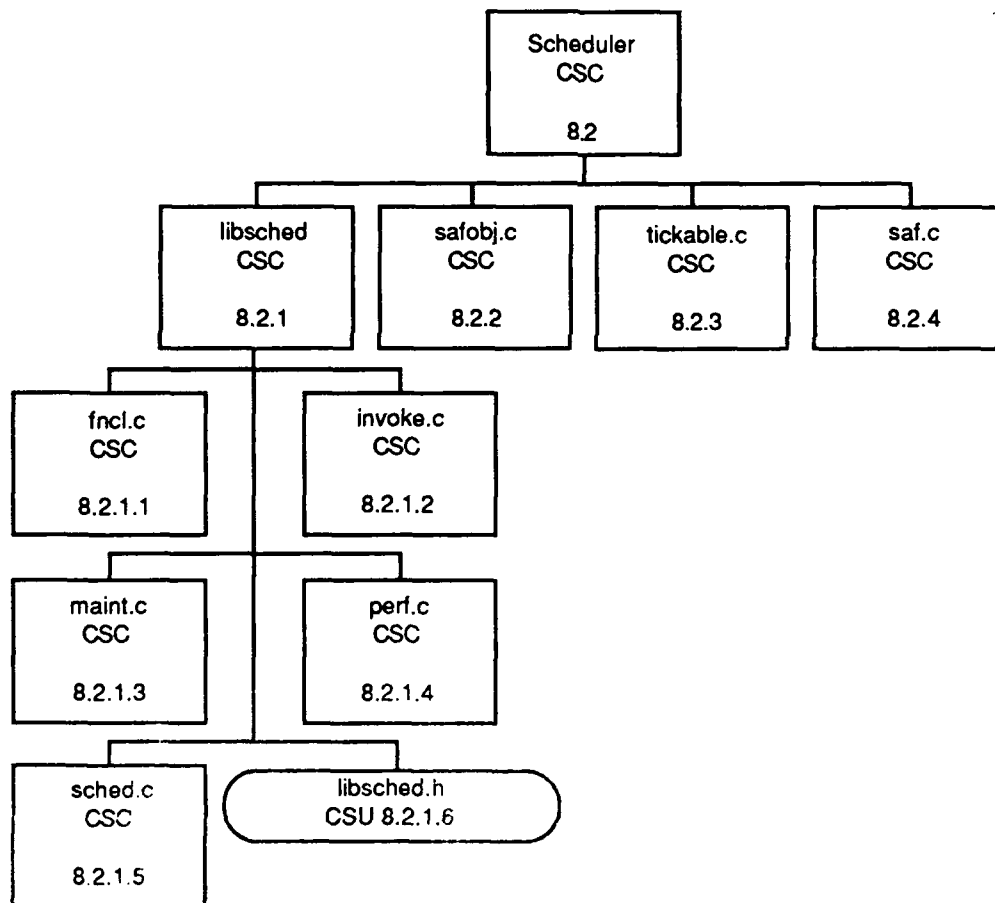


Figure 2.2-1: SCHEDULER CSC Structure

2.2.1 libsched CSC

/simnet/libsrc/libsched

This library handles the scheduler, which calls functions when their time to execute occurs. Whether a function is called is determined by checking the time for the next execution of the function in the scheduler rings. If the current "clock" time is past that scheduled time, the function is executed.

2.2.1.1 fncl.c CSC

/simnet/libsrc/libsched/fncl.c

This CSC contains scheduler entry point CSUs. Its structure is depicted in Figure 2.2-2.

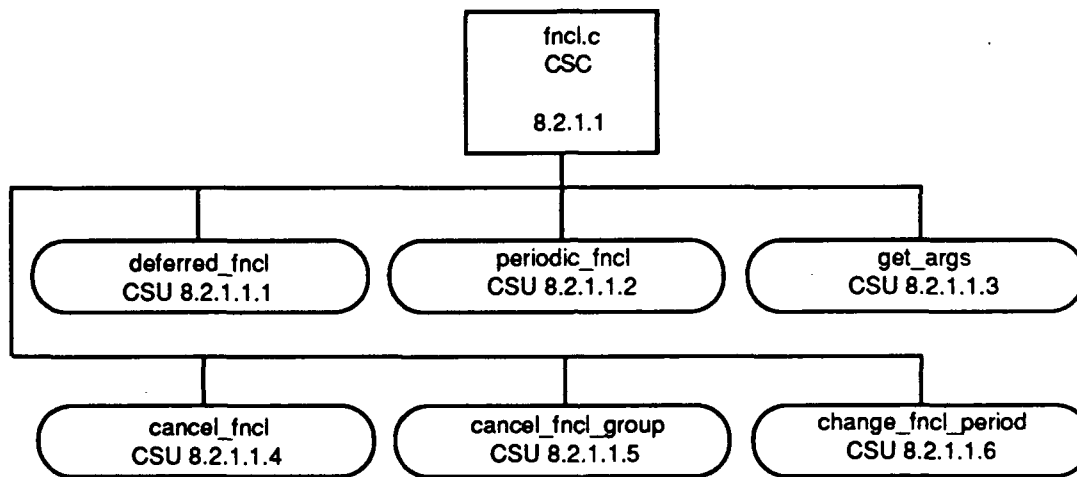


Figure 2.2-2: fncl.c CSC Structure

2.2.1.1.1 deferred_fncl CSU

This CSU initializes a function which is delayed by the period of time passed in the "delay" parameter and then executed once.

Parameters		
Parameters	Type	Where Typedef Declared
func	FUNC PTR	
delay	unsigned int	Standard
group	int	Standard
Calls		
Function	Where Described	
heap allocate	Sec. 2.14.2.2.1	
get millisecond time	Sec. 2.14.3.3.2	
get args	Sec. 2.2.1.1.3	
insert function	Sec. 2.2.1.3.5	

Table 2.2-1: deferred_fncl CSU [8.2.1.1.1]

2.2.1.1.2 periodic_fncl CSU

This CSU is similar to deferred_fncl, except that after the initial delay the function is executed on a periodic basis.

Parameters		
Parameters	Type	Where Typedef Declared
func	FUNC_PTR	
delay	unsigned int	Standard
group	int	Standard
Calls		
Function	Where Described	
heap_allocate	Sec. 2.14.2.2.1	
get_millisecond_time	Sec. 2.14.3.3.2	
get_args	Sec. 2.2.1.1.3	
insert_function	Sec. 2.2.1.3.5	

Table 2.2-2: periodic_fncl CSU [8.2.1.1.2]

2.2.1.1.3 get_args CSU

This CSU, whose parameters consist of a pointer to a functional description structure and a list of arguments, checks variable argument type for (and as long as there are) integers or doubles and not the end of the list (or an unknown argument type). On each successful check, it increments the count. Upon completion, it sets the structure's argument count, gets memory to store the arguments, storing a pointer to the memory, and copies the arguments into memory.

Parameters		
Parameters	Type	Where Typedef Declared
fd	pointer to FUNCTION_DESCRIPTION	Sec. 2.2.1.6
Calls		
Function	Where Described	
heap_allocate	Sec. 2.14.2.1.1	

Table 2.2-3: get_args CSU [8.2.1.1.3]

2.2.1.1.4 cancel_fncl CSU

This CSU removes a functional description through a call to remove_function. If the item is currently executing, free_when_done is set, but if it is not executing, the function is freed through a call to free_function.

Parameters		
Parameters	Type	Where Typedef Declared
id	int	Standard

Calls	
Function	Where Described
remove_function	Sec. 2.2.1.3.3
free_function	Sec. 2.2.1.3.2

Table 2.2-4: cancel_fncl CSU [8.2.1.1.4]

2.2.1.1.5 cancel_fncl_group CSU

This CSU collects a list of thing to cancel, first the event rings and then the functional description groups, and then cancels them through a call to cancel_fncl.

Parameters		
Parameters	Type	Where Typedef Declared
group	int	Standard
Calls		
Function	Where Described	
cancel_fncl	Sec. 2.2.1.1.4	

Table 2.2-5: cancel_fncl_group CSU [8.2.1.1.5]

2.2.1.1.6 change_fncl_period CSU

This CSU removes from the scheduler the function whose id is passed. It sets the function's period to the new_period, and sets the time that the operation is to take place. It then inserts the function back into the scheduler.

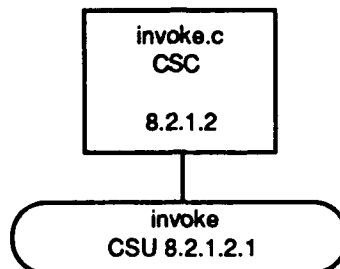
Parameters		
Parameters	Type	Where Typedef Declared
id	int	Standard
new_period	unsigned int	Standard
Calls		
Function	Where Described	
remove_function	Sec. 2.2.1.3.3	
get_millisecond_time	Sec. 2.14.3.3.2	
insert_function	Sec. 2.2.1.3.5	

Table 2.2-6: change_fncl_period CSU [8.2.1.1.6]

2.2.1.2 invoke.c CSC

/simnet/libsrc/libsched/invoke.c

This CSC contains only one CSU, invoke, as shown in Figure 2.2-3.

**Figure 2.2-3: invoke.c CSC Structure****2.2.1.2.1 invoke CSU**

This CSU is the only one within the CSC invoke.c. Its purpose is to invoke a function, the pointer to which being passed as a parameter. Within the function are five macro definitions, C1(t0), C2(t0,t1), C3(t0,t1,t2), C4(t0,t1,t2,t3), and S(n,x,y), which are defined in Appendix A.

Parameters		
Parameters	Type	Where Typedef Declared
fd	pointer to FUNCTION DESCRIPTION	Sec. 2.2.1.6
Calls		
Function	Where Described	
S	This section See Appendix A	
C1	This section See Appendix A	
C2	This section See Appendix A	
C3	This section See Appendix A	
C4	This section See Appendix A	
free function	Sec. 2.2.1.3.2	

Table 2.2-7: invoke CSU [8.2.1.2.1]**2.2.1.3 maint.c CSC**

/simnet/libsrc/libsched/maint.c

This CSC consists of six CSUs required for scheduler maintenance.

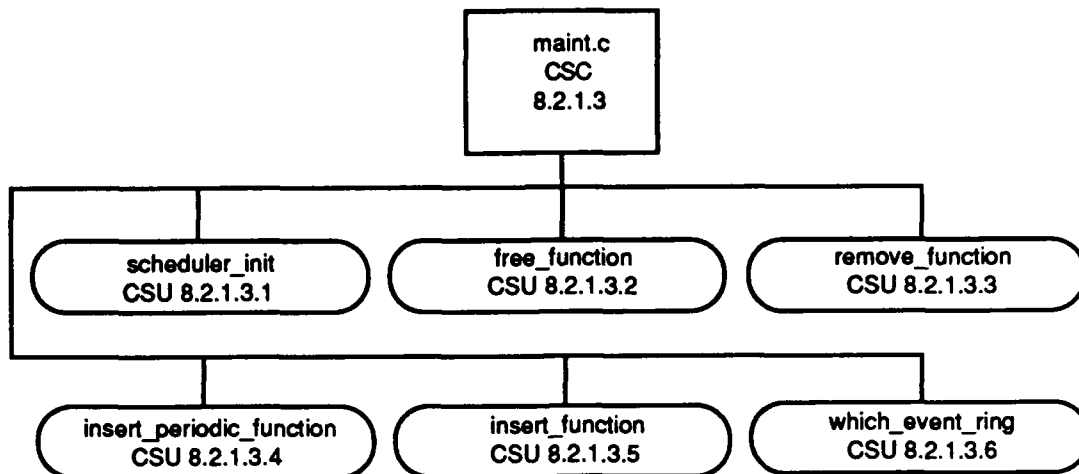


Figure 2.2-4: maint.c CSC Structure

2.2.1.3.1 scheduler_init CSU

This CSU gets and sets up function description structures for the number of rings passed, as long as the number is less than the maximum number of rings allowed. It then sets both the event list head and tail to zero since neither position has yet been filled.

Parameters		
Parameters	Type	Where Typedef Declared
nrings	int	Standard
Calls		
Function	Where Described	
heap allocate	Sec. 2.14.2.1.1	

Table 2.2-8: scheduler_init CSU [8.2.1.3.1]

2.2.1.3.2 free_function CSU

This CSU frees the memory heap used by the function arguments and then frees the heap used by the function.

Parameters		
Parameters	Type	Where Typedef Declared
fd	pointer to FUNCTION DESCRIPTION	Sec. 2.2.1.6
Calls		
Function	Where Described	
heap deallocate	Sec. 2.14.2.1.4	

Table 2.2-9: free_function CSU [8.2.1.3.2]

2.2.1.3.3 remove_function CSU

This CSU removes the function from either the event list or the event rings. If the function was the last function in the lowest (fastest period) occupied ring, then by definition the lowest occupied ring must be a higher ring. The CSU then find the ring. An error check is done at the end to ensure integrity of the scheduler data structures.

Parameters		
Parameters	Type	Where Typedef Declared
id	pointer to FUNCTION DESCRIPTION	Sec. 2.2.1.6
Calls		
Function	Where Described	
which event ring	Sec. 2.2.1.3.6	

Table 2.2-10: remove_function CSU [8.2.1.3.3]

2.2.1.3.4 insert_periodic_function CSU

This CSU determines into which event ring the function should be placed, and sets the function description structure accordingly. If this function is going below the current lowest occupied ring, the new lowest occupied ring is set to the ring that the function occupies.

Parameters		
Parameters	Type	Where Typedef Declared
id	pointer to FUNCTION DESCRIPTION	Sec. 2.2.1.6
Calls		
Function	Where Described	
which event ring	Sec. 2.2.1.3.6	

Table 2.2-11: insert_periodic_function CSU [8.2.1.3.4]

2.2.1.3.5 insert_function CSU

This CSU inserts a function into the scheduler event list determined by when the function should be scheduled.

Parameters		
Parameters	Type	Where Typedef Declared
id	pointer to FUNCTION DESCRIPTION	Sec. 2.2.1.6

Table 2.2-12: insert_function CSU [8.2.1.3.5]

2.2.1.3.6 which_event_ring CSU

This CSU returns the event ring for the period passed, if one exists.

Parameters		
Parameters	Type	Where Typedef Declared
period	unsigned int	Standard
ReturnValues		
Return Value	Type	Meaning
i	unsigned short	Event ring for period passed
0	unsigned short	No event ring for period

Table 2.2-13: which_event_ring CSU [8.2.1.3.6]

2.2.1.4 perf.c CSC

/simnet/libsrc/libsched/perf.c

This CSC contains six CSUs required to monitor system performance. Its structure is shown in Figure 2.2-5.

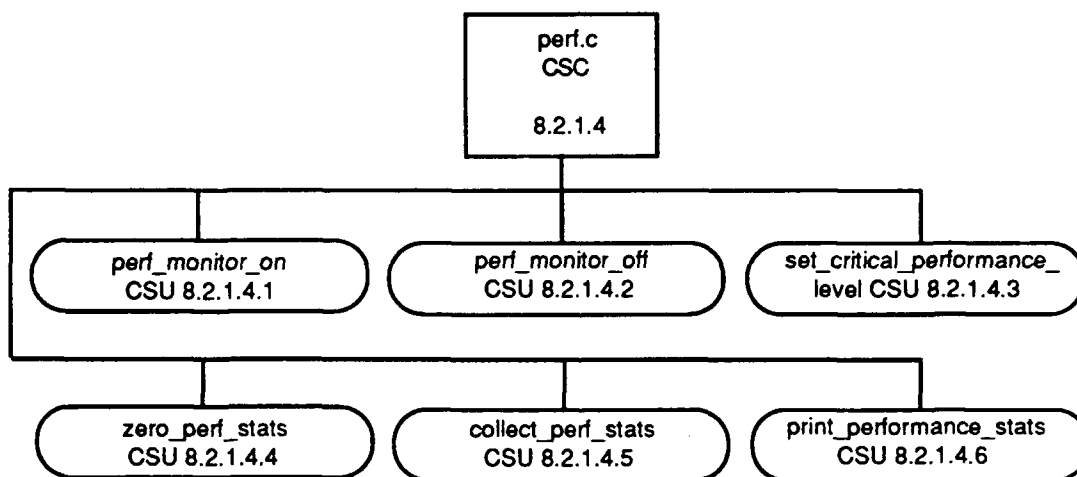


Figure 2.2-5: perf.c CSC Structure

2.2.1.4.1 perf_monitor_on CSU

This CSU turns the performance monitoring functions on, setting the performance monitoring period to the period passed, getting the clock time, and zeroing out the previous performance statistics.

Parameters		
Parameters	Type	Where Typedef Declared
period	unsigned int	Standard

Calls	
Function	Where Described
zero_performance_stats	Sec. 2.2.1.4.4
get_millisecond_time	Sec. 2.14.3.3.2

Table 2.2-14: perf_monitor_on CSU [8.2.1.4.1]

2.2.1.4.2 perf_monitor_off CSU

This CSU turns the performance monitor off by setting the performance monitor period to zero.

2.2.1.4.3 set_critical_performance_level CSU

This CSU sets the critical performance levels for monitoring system performance by setting these static variables to the values passed: to zero in the cases of critical_count and critical_total, to the critical_ring (based on a call to which_event_ring, passing the period), and critical_next_check (a function of the present time plus for_how_long).

Parameters		
Parameters	Type	Where Typedef Declared
period	unsigned int	Standard
ratio	double	Standard
under_threshold	unsigned int	Standard
for how long	unsigned int	Standard
stress function	FUNC PTR	
relief function	FUNC PTR	
Calls		
Function	Where Described	
which_event_ring	Sec. 2.2.1.3.6	
get_millisecond	Sec. 2.14.3.3.2	

Table 2.2-15: set_critical_performance_level CSU [8.2.1.4.3]

2.2.1.4.4 zero_perf_stats CSU

This CSU sets all the performance statistics to zero, sets the last performance check for each ring to zero, and sets next print time of the performance monitor (perf_monitor_next_print) to the sum of the current clock time (passed as a parameter) plus the performance monitoring period (perf_monitor_period).

Parameters		
Parameters	Type	Where Typedef Declared
clock	unsigned int	Standard

Table 2.2-16: zero_perf_stats CSU [8.2.1.4.4]

2.2.1.4.5 collect_perf_stat CSU

This CSU collects performance statistics. It first checks if it printed the statistics on the last performance check, and if so, sets that flag to "clock", indicating that the skipped cycle has occurred. If the clock is past the time of waiting to print the statistics, the routine prints the statistics and zeroes the performance statistics. Depending on the ring passed (equivalent to the critical ring), the clock (past the next critical check), and the resulting critical_count and critical_total, the critical_state may be complemented (zero to one or one to zero). Depending on the critical_ratio, the critical_stress_function or the critical_relief_function may be called. The CSU updates the time for the next critical check and sets both critical_count and critical_total to zero.

Parameters		
Parameters	Type	Where Typedef Declared
ring	unsigned short	Standard
clock	unsigned int	Standard
Calls		
Function	Where Described	
min	sim_macros.h	
print_perf_stats	Sec. 2.2.1.4.6	
zero_perf_stats	Sec. 2.2.1.4.4	
critical_stress_function		
critical_relief_function		

Table 2.2-17: collect_perf_stat CSU [8.2.1.4.5]

2.2.1.4.6 print_performance_stats CSU

This CSU prints the collected performance monitoring statistics.

2.2.1.5 sched.c CSC

/simnet/libsrc/libsched/sched.c

In addition to the two functions shown in Figure 2.2-6, the file contains four event ring and two event list declarations and a macro (min3(a,b,c)) described in Appendix A.

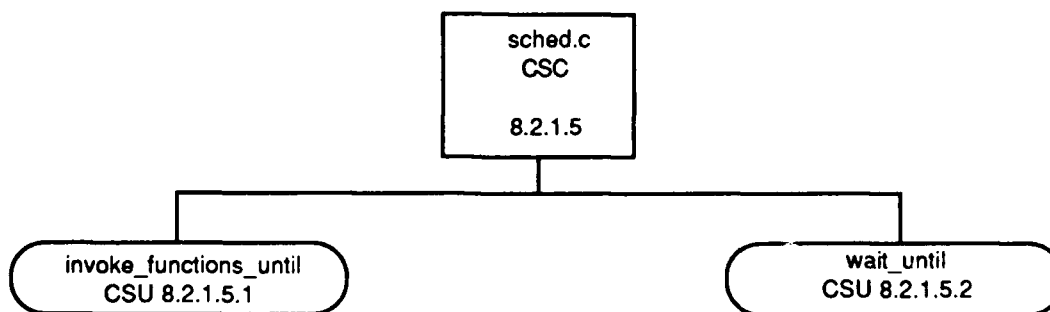


Figure 2.2-6: sched.c CSC Structure

2.2.1.5.1 invoke_functions_until CSU

This CSU invokes functions in the scheduler rings until the deadline (passed in the input parameter) occurs.

Parameters		
Parameters	Type	Where Typedef Declared
deadline	unsigned int	Standard
Calls		
Function	Where Described	
get_millisecond_time	Sec. 2.14.3.3.2	
insert_periodic_function	Sec. 2.2.1.3.4	
invoke	Sec. 2.2.1.2.1	
collect_perf_stat	Sec. 2.2.1.4.5	
min3	Sec. 2.2.1.5 See Appendix A	
invoke_functions_until	Sec. 2.2.1.5.1	
wait_until	Sec. 2.2.1.5.2	

Table 2.2-18: invoke_functions_until CSU [8.2.1.5.1]

2.2.1.5.2 wait_until CSU

This CSU waits until the time is at or past the deadline.

Parameters		
Parameters	Type	Where Typedef Declared
deadline	unsigned int	Standard
Calls		
Function	Where Described	
get_millisecond_time	Sec. 2.14.3.3.2	

Table 2.2-19: wait_until CSU [8.2.1.5.2]

2.2.1.6 libsched.h CSU

/simnet/libsrc/libsched/libsched.h

This include file, in addition to a number of external declarations, contains a number of constant defines and two structure definitions (typedef struct ARG_UNION and FUNCTION_DESCRIPTION), which are described in the following tables.

Constant	Value
MAX RINGS	8
PERF WIDTH	20
A END	0 /* Argument types */
A INT	1
A DOUBLE	2
A CHAR	1 /* Everything else promoted to int */
A SHORT	1
A FLOAT	2 /* or double */
A PTR	1
ASAP	0 /* no delay */

Table 2.2-20: `libsched.h` Constants

Item	Type	Where Type Defined
tag	unsigned short	Standard
union		
{		
int arg	int	Standard
double arg	double	Standard
} arg		

Table 2.2-21: ARG_UNION Structure Definition

Item	Type	Where Type Defined
when	unsigned int	Standard
func	FUNC_PTR	
period	unsigned int	Standard
marker :1	unsigned int	Standard
executing :1	unsigned int	Standard
free when done :1	unsigned int	Standard
periodic :1	unsigned int	Standard
arg_count	short	Standard
group	int	Standard
args	pointer to ARG_UNION	Above
prev	pointer to struct function_description	Here, pointer to previous structure
next	pointer to struct function_description	Here, pointer to next structure

Table 2.2-22: FUNCTION_DESCRIPTION Structure Definition

2.2.2 safobj.c CSC

/simnet/src/host/safobj.c

This CSC contains the CSUs to construct the object table which is used to track all of the locally simulated vehicles, remote vehicles, and units.

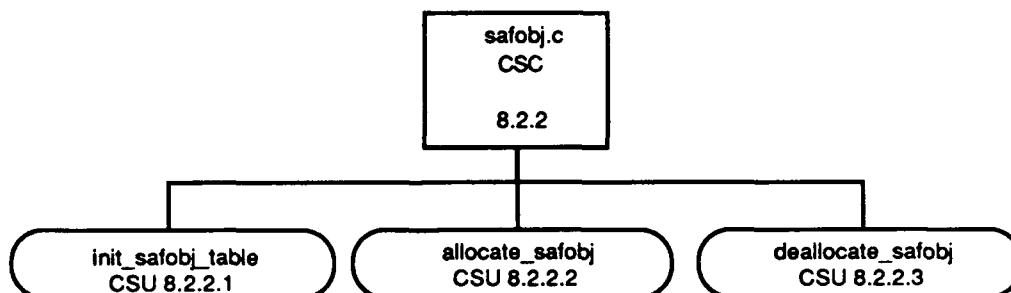


Figure 2.2-7: safobj.c CSC Structure

2.2.2.1 init_safobj_table CSU

This CSU initializes a SAF object table by setting all elements to zero.

2.2.2.2 allocate_safobj CSU

This CSU allocates a SAF object table, if a check of the id passed does not already show it in use. It does so by allocating memory for it, zeroing out the memory, and assigning the SAF type passed to the saf_type for that id in the SAF object table (g_safobj_table[id] -> saf_type).

Parameters		
Parameters	Type	Where Typedef Declared
id	unsigned int	Standard
saf_type	unsigned short	Standard
ReturnValues		
Return Value	Type	Meaning
g_safobj_table[id]	pointer to SAF_OBJECT	Pointer to the table
Errors		
Error Name	Reason for Error	
ERROR_OUT	The allocated SAF object is already in use	
Calls		
Function	Where Described	
heap_allocate	Sec 2.14.2.1.1	
ERROR_OUT	Sec. 2.5.2.2	

Table 2.2-23: allocate_safobj CSU [8.2.2.2]

2.2.2.3 deallocate_safobj CSU

This CSU releases the SAF object memory and sets its table identification to NULL.

Parameters		
Parameters	Type	Where Typedef Declared
id	unsigned int	Standard
Calls		
Function	Where Described	
heap_deallocate	Sec. 2.14.2.1.4	

Table 2.2-24: deallocate_safobj CSU [8.2.2.3]

2.2.3 tickable.c CSC

/simnet/src/host/tickable.c

The tickable.c CSC structure is shown in Figure 2.2-8. This CSC contains the CSUs to create and delete the TICKABLE_VARS structure with all of the vehicle ticking variables. It also contains the CSUs to cause a vehicle to start and stop ticking. The way a vehicle is simulated is by updating its state at periodic intervals, called ticks. At this time, the program checks to insure that a state change is not needed in any facet of the simulation. If a state change is needed, the program accomplishes this task. Appearance packets are sent at this time as needed. A vehicle's tick rate (how often its condition is updated and displayed) can be adjusted for various reasons, one of which being destruction. (Dead vehicles tick much less frequently than live ones, since they do nothing but send out appearance packets.)

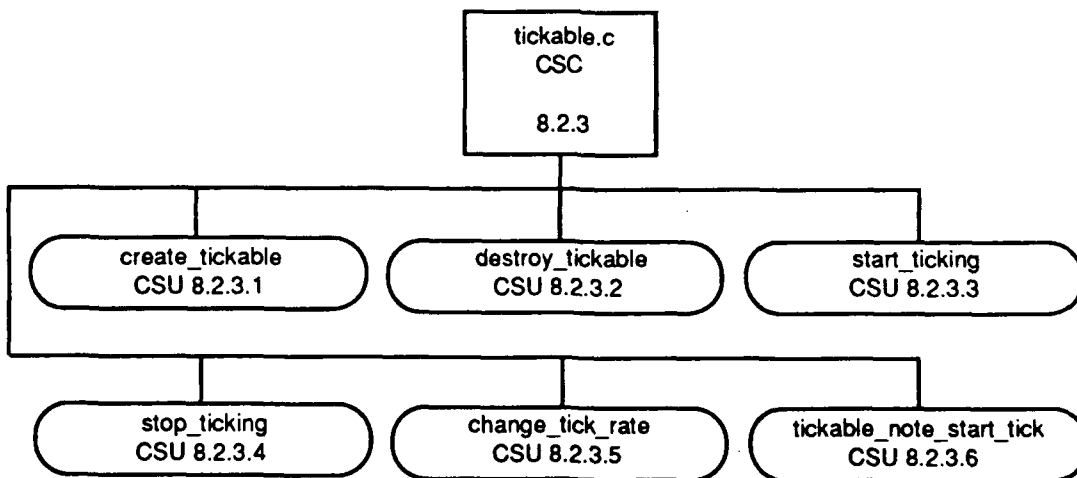


Figure 2.2-8: tickable.c CSC Structure

2.2.3.1 create_tickable CSU

This CSU causes a memory heap to build a periodic interval (tickable) and initializes it (based on the parameters passed), returning a pointer to the TICKABLE_VARS structure.

Parameters		
Parameters	Type	Where Typedef Declared
tick function	FUNC PTR	
period	unsigned int	Standard
ReturnValues		
Return Value	Type	Meaning
tickable	pointer to TICKABLE_VARS	Points to memory structure
Calls		
Function	Where Described	
allocate_tickable	Sec. 2.9.1.2 See Appendix A	
get_millisecond_time	Sec. 2.14.3.3.2	

Table 2.2-25: create_tickable CSU [8.2.3.1]

2.2.3.2 destroy_tickable CSU

This CSU destroys a tickable by releasing its memory heap.

Parameters		
Parameters	Type	Where Typedef Declared
tickable	pointer to TICKABLE_VARS	
Calls		
Function	Where Described	
deallocate_tickable	Sec. 2.9.1.2 See Appendix A	

Table 2.2-26: destroy_tickable CSU [8.2.3.2]

2.2.3.3 start_ticking CSU

This CSU causes a SAF object to start ticking through a call to periodic_fnc1.

Parameters		
Parameters	Type	Where Typedef Declared
safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
periodic_fnc1	Sec. 2.2.1.1.2	

Table 2.2-27: start_ticking CSU [8.2.3.3]

2.2.3.4 stop_ticking CSU

This CSU causes a SAF object to stop ticking through a call to `cancel_fncl`.

Parameters		
Parameters	Type	Where Typedef Declared
<code>safobj</code>	pointer to SAF_OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
<code>cancel_fncl</code>	Sec. 2.2.1.1.4	

Table 2.2-28: stop_ticking CSU [8.2.3.4]

2.2.3.5 change_tick_rate CSU

This CSU changes a SAF object's tick rate by first changing the object's tickable period in milliseconds to the value passed in the period parameter and then calling `change_fncl_period`.

Parameters		
Parameters	Type	Where Typedef Declared
<code>safobj</code>	pointer to SAF_OBJECT	Sec. 2.9.1.1
<code>period</code>	unsigned int	Standard
Calls		
Function	Where Described	
<code>change_fncl_period</code>	Sec. 2.2.1.1.6	

Table 2.2-29: change_tick_rate CSU [8.2.3.5]

2.2.3.6 tickable_note_start_tick CSU

This CSU notes the start of the tick by obtaining the tickable time since the last tick and updating the current time (`tickable > now`) to the last millisecond time.

Parameters		
Parameters	Type	Where Typedef Declared
<code>tickable</code>	pointer to TICKABLE_VARS	Sec. 2.9.1.2

Table 2.2-30: tickable_note_start_tick CSU [8.2.3.6]

2.2.4 saf.c CSC

/simnet/src/host/saf.c

This CSC contains the code to reset the Simhost CSCI (phantom program) and to clear the vehicles currently being simulated. Figure 2.2-9 presents the saf.c CSC structure.

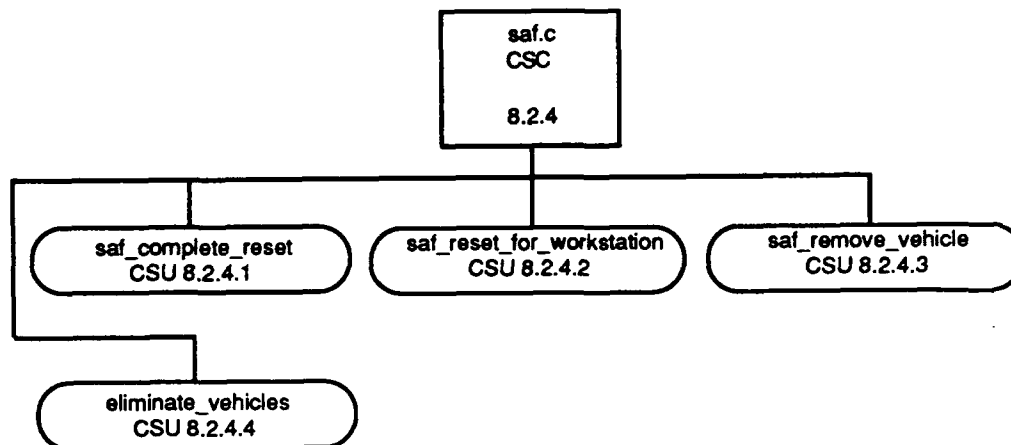


Figure 2.2-9: saf.c CSC Structure

2.2.4.1 saf_complete_reset CSU

This CSU completely resets all current SAF vehicles by checking each SAF object for its existence, and then eliminating the vehicle if it does exist.

Calls	
Function	Where Described
LOOKUP SAFOBJ	Sec. 2.9.1.1 See Appendix A
eliminate_vehicles	Sec. 2.2.4.4

Table 2.2-31: saf_complete_reset CSU [8.2.4.1]

2.2.4.2 saf_reset_for_workstation CSU

This CSU resets all current SAF vehicles used by the requesting workstation by checking for the existence of each of the workstation's SAF objects, and then eliminating the existing vehicles.

Parameters		
Parameters	Type	Where Typedef Declared
sbx	pointer to SBX_CONNECTION_VARS	Sec 2.4.3.3

Calls	
Function	Where Described
LOOKUP_SAFOBJ	Sec. 2.9.1.1 See Appendix A
OBJ_OWNER_PORT_NUMBER	Sec. 2.9.1.1 See Appendix A
eliminate_vehicles	Sec. 2.2.4.4

Table 2.2-32: saf_reset_for_workstation CSU [8.2.4.2]

2.2.4.3 saf_remove_vehicle CSU

This CSU removes a vehicle, whose identification number is passed as the sole parameter.

Parameters		
Parameters	Type	Where Typedef Declared
id	unsigned int	Standard
Calls		
Function	Where Described	
eliminate_vehicles	Sec. 2.2.4.4	

Table 2.2-33: saf_remove_vehicle CSU [8.2.4.3]

2.2.4.4 eliminate_vehicles

This CSU eliminates the number of vehicles passed in the vehicle list, which is also passed. It searches for each vehicle, and informs the world that it is disappearing via a call to remove_vehicles. It then searches for each vehicle again on the vehicle list, and if the vehicle does exist, it removes it through a call to go_away. If the vehicle does not exist, it sends an error message to the terminal via ERROR_OUT.

Parameters		
Parameters	Type	Where Typedef Declared
num_v	int	Standard
v_list[]	unsigned int	Standard
Errors		
Error Name	Reason for Error	
ERROR_OUT	Attempted to eliminate a vehicle that doesn't exist.	
Calls		
Function	Where Described	
LOOKUP_SAFOBJ	Sec. 2.9.1.1 See Appendix A	
remove_vehicles	Sec. 2.14.1.1.5	
go_away	Sec. 2.14.1.1.6	
ERROR_OUT	Sec. 2.5.2.2	

Table 2.2-34: eliminate_vehicles CSU [8.2.4.4]

2.3 SIMNET INTERFACE CSC

Code running on an intelligent Ethernet controller receives SIMNET protocol packets from the network, filters them, and places them on a queue. When the SIMNET interface code on the main processor gets ticked (called periodically) by the Scheduler CSC, it copies these packets into a second queue and places the addresses of these packets on the queues of appropriate Local Vehicles, Remote Vehicles, Units, and SAF Command Interface instantiations so that these packets can be processed when the CSC is ticked. The SIMNET Interface CSC gets called directly by these CSC instantiations when the SAF Simhost CSCI is to send out SIMNET packets. These calls take place in the thread of the CSC's ticks, not the SIMNET Interface CSC's tick.

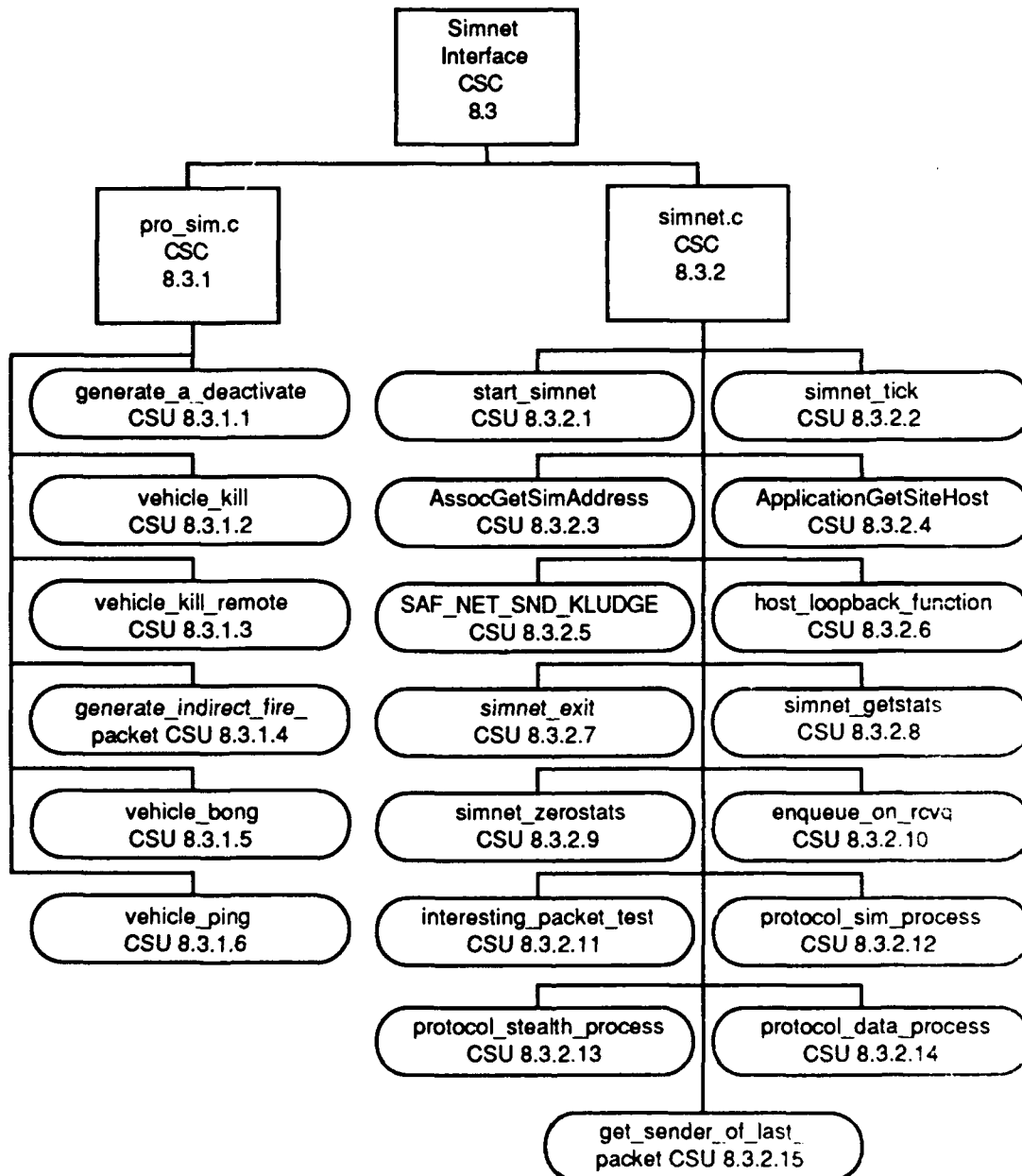


Figure 2.3-1: Simnet Interface CSC Structure

2.3.1 pro_sim.c CSC

/simnet/src/host/pro_sim.c

This CSC handles sending out fire and damage packets to the other vehicles on the network.

2.3.1.1 generate_a_deactivate CSU

This CSU calls simnet_send_deactivate if the vehicle is some form of local simulator type.

Parameters		
Parameters	Type	Where Typedef Declared
id	unsigned int	Standard
Calls		
Function	Where Described	
LOOKUP VEHICLES	Sec. 2.9.3.2	
simnet send deactivate		
ERROR OUT	Sec. 2.5.2.2	

Table 2.3-1: generate_a_deactivate CSU [8.3.1.1]

2.3.1.2 vehicle_kill CSU

How this CSU kills a vehicle depends on whether the vehicle is local or remote. If local, a call is made to saf_vehicle_catastrophic_kill. If the vehicle is remote, a call is made to vehicle_kill_remote. If the vehicle is neither, an error message is sent indicating that the vehicle with that id can not be located.

Parameters		
Parameters	Type	Where Typedef Declared
id	unsigned int	Standard
Calls		
Function	Where Described	
LOOKUP SAFOBJ	Sec. 2.9.1.1 See Appendix A	
SAF LOCAL VEHICLE	Sec. 2.9.3.2 See Appendix A	
saf_vehicle_catastrophic_kill	Sec. 2.6.1.1.25	
SAF REMOTE VEHICLE	Sec. 2.9.3.2 See Appendix A	
vehicle_kill_remote	Sec. 2.3.1.3	
ERROR OUT	Sec. 2.5.2.2	

Table 2.3-2: vehicle_kill CSU [8.3.1.2]

2.3.1.3 vehicle_kill_remote CSU

This CSU kills a remote vehicle by getting the vehicle's position and calling `simnet_send_impact` to send a lethal hit.

Parameters		
Parameters	Type	Where Typedef Declared
id	unsigned int	Standard
Calls		
Function	Where Described	
LOOKUP_POSITION	Sec. 2.9.1.1 See Appendix A	
simnet_send_impact		

Table 2.3-3: vehicle_kill_remote CSU [8.3.1.3]

2.3.1.4 generate_indirect_fire_packet CSU

This CSU checks the artillery type, and, if it is an existing type (vehicle or ground) and the vehicle target exists, determines hit position and calls `simnet_send_indirect_fire`.

Parameters		
Parameters	Type	Where Typedef Declared
mp	pointer to ARTY_MSG	Sec. 2.4.1.1
Calls		
Function	Where Described	
LOOKUP_VEHICLE	Sec. 2.9.3.2	
ERROR_OUT	Sec. 2.5.2.2	
vec_copy	Sec. 2.6.2.59.1 in Vehicles CSCI SDD	
tdb_get_gl	Sec. 2.14.1.2.2	
simnet_send_indirect_fire		

Table 2.3-4: generate_indirect_fire_packet CSU [8.3.1.4]

2.3.1.5 vehicle_bong CSU

This CSU applies a deadly vehicle impact by searching for the remote vehicle and, if found, calling `vehicle_kill_remote`.

Parameters		
Parameters	Type	Where Typedef Declared
victim_id	unsigned int	Standard
Calls		
Function	Where Described	
LOOKUP_VEHICLE	Sec. 2.9.3.2	
vehicle_kill_remote	Sec. 2.3.1.3	

Table 2.3-5: vehicle_bong CSU [8.3.1.5]

2.3.1.6 vehicle_ping CSU

This CSU applies a mild vehicle impact from the parser by searching for the remote vehicle and, if it is found, acquiring the vehicle position and calling `simnet_send_impact`.

Parameters		
Parameters	Type	Where Typedef Declared
<code>victim_id</code>	unsigned int	Standard
Calls		
Function	Where Described	
<code>LOOKUP_VEHICLE</code>	Sec. 2.9.3.2	
<code>LOOKUP_POSITION</code>	Sec. 2.9.1.1	
<code>simnet_send_impact</code>		

Table 2.3-6: vehicle_ping CSU [8.3.1.6]

2.3.2 simnet.c CSC

/simnet/src/host/simnet.c

This CSC handles receiving, queuing and distributing relevant SIMNET simulation packets received from the network. In addition to the CSUs, there is a single constant define called `NUM_SIMNET_BUFS` setting the number of SIMNET buffers, and two string definitions, as shown in the following two tables.

Constant	Value
<code>NUM_SIMNET_BUFS</code>	4096

Table 2.3-7: NUM_SIMNET_BUFS Constant Definition

Pointer Name	String
<code>char *enp_device =</code>	<code>"/dev/enp0"</code>
<code>char *g_netstart_args[] =</code>	<code>{"saf/bin/netstart", 0}</code>

Table 2.3-8: simnet.c String Definitions

2.3.2.1 start_simnet CSU

This CSU performs all the tasks required to start the SIMNET Interface.

Calls	
Function	Where Described
buffer pool allocate	Sec. 2.14.4.1.13
PointToPointOpen	Sec. 2.1.1.2.2.1.1 in Vehicles CSCI SDD
ERROR_OUT	Sec. 2.5.2.2
AssocError	Sec. 2.20.1.10.11 in MCC CSCI SDD
exit all sbx conns	Sec. 2.4.3.2.2
simnet_exit	Sec. 2.3.2.7
fork	
execve	
AssocSubscrib	Sec. 2.20.1.10.11 in MCC CSCI SDD
periodic fncl	Sec. 2.2.1.1.2
simnet_zerostats	Sec. 2.3.2.9

Table 2.3-9: start_simnet CSU [8.3.2.1]

2.3.2.2 simnet_tick CSU

This CSU performs all the tasks required for each SIMNET tick.

Errors	
Error Name	Reason for Error
ERROR_OUT	Error from association layer on tick
ERROR_OUT	Received packet of unknown protocol
Calls	
Function	Where Described
AssocTickAssocLayer	Sec. 2.20.1.8.1 in MCC CSCI SDD
ERROR_OUT	Sec. 2.3.2.2
AssocError	Sec. 2.20.1.10.11 in MCC CSCI SDD
PointToPointReceivePDU	Sec. 2.1.1.2.2.2.1 in Vehicles CSCI SDD
AssocSendResponse	Sec. 2.20.1.4.2 in MCC CSCI SDD
interesting packet test	Sec. 2.3.2.11
buffer allocate from pool	Sec. 2.14.4.1.10
protocol sim process	Sec. 2.3.2.12
buffer deallocate	Sec. 2.14.4.2.14
buffer allocate	Sec. 2.14.4.2.12
protocol stealth process	Sec. 2.3.2.13
get sender of last packet	Sec. 2.3.2.15

Table 2.3-10: simnet_tick CSU [8.3.2.2]

2.3.2.3 AssocGetSimAddress CSU

This CSU sets the site and host elements of the passed simulation address structure to `g_site_number` and `g_host_number`, respectively.

Parameters		
Parameters	Type	Where Typedef Declared
handle	int	Standard
simAddress	pointer to SimulationAddress	address.h

Table 2.3-11: AssocGetSimAddress CSU [8.3.2.3]

2.3.2.4 ApplicationGetSiteHost CSU

This CSU sets the site pointer to `g_site_number` and the host pointer to `g_host_number`.

Parameters		
Parameters	Type	Where Typedef Declared
handle	int	Standard
site	pointer to short	Standard
host	pointer to short	Standard

Table 2.3-12: ApplicationGetSiteHost CSU [8.3.2.4]

Prior to CSU `SAF_NET_SND_KLUDGE`, `g_ethernet_two_packets`, which determines which packet type is sent, is set to FALSE and the following header values are defined.

```
static NetworkHeader hdr = { { 0xff, 0xff, 0xff, 0xff, 0xff },
                             { 0xff, 0xff, 0xff, 0xff, 0xff },
                             21000 },
```

2.3.2.5 SAF_NET_SND_KLUDGE CSU

This CSU allows the sending of Ethernet 2.0 packets.

Parameters		
Parameters	Type	Where Typedef Declared
h	int	Standard
to	pointer to NetworkAddress	network.h
buf	pointer to char	Standard
len	int	Standard
flags	int	Standard
ReturnValues		
Return Value	Type	Meaning
net_send(h, to, buf, len, flags)	int	Not 2.0 packets
net_send(h, &hdr, buf, len, flags)	int	2.0 packets, bcopy required before this return

Calls	
Function	Where Described
net_send	Sec. 2.20.2.15.1 in MCC CSCI SDD

Table 2.3-13: SAF_NET_SND_KLUDGE CSU [8.3.2.5]

2.3.2.6 host_loopback_function CSU

This CSU determines if a looped back packet is of interest. If the protocol number indicates a simulation, protocol_sim_process is called. If the protocol number indicates a collection of data, protocol_data_process is called. If the protocol is neither, ERROR_OUT is called informing the user that the packet protocol is unknown.

Parameters		
Parameters	Type	Where Typedef Declared
data	pointer to char	Standard
length	long int	Standard
group	MulticastGroupID	p_assoc.h
protocol	AssociationUserProtocol	p_assoc.h
Errors		
Error Name	Reason for Error	
ERROR_OUT	Unknown packet type	
Calls		
Function	Where Described	
interesting_packet_test	Sec. 2.3.2.11	
protocol sin, process	Sec. 2.3.2.12	
protocol_data_process	Sec. 2.3.2.14	
ERROR_OUT	Sec. 2.5.2.2	

Table 2.3-14: host_loopback_function CSU [8.3.2.6]

2.3.2.7 simnet_exit CSU

This CSU calls AssocClose, passing the SIMNET handle as a parameter.

Calls	
Function	Where Described
AssocClose	Sec. 2.20.1.11.1 in MCC CSCI SDD

Table 2.3-15: simnet_exit CSU [8.3.2.7]

2.3.2.8 simnet_getstats CSU

This CSU gets the SIMNET statistics and prints them.

Parameters		
Parameters	Type	Where Typedef Declared
stats[100]	long	Standard
s[41]	char	Standard
i	register int	Standard
Calls		
Function	Where Described	
net_get_statistics	Sec. 2.20.2.5.1 in MCC CSCI SDD	
net_stat_string	Sec. 2.20.2.5.3 in MCC CSCI SDD	

Table 2.3-16: simnet_getstats CSU [8.3.2.8]

2.3.2.9 simnet_zerostats CSU

This CSU initializes or sets to zero the SIMNET statistics.

Calls	
Function	Where Described
net_zero_statistics	Sec. 2.20.2.5.2 in MCC CSCI SDD

Table 2.3-17: simnet_zerostats CSU [8.2.1.2.9]

2.3.2.10 enqueue_on_rcvq CSU

This CSU places a simple received SAF object packet on the receive queue if the queue is not out of space.

Parameters		
Parameters	Type	Where Typedef Declared
id	unsigned int	Standard
pdu	pointer to char	Standard
type	int	Standard
Calls		
Function	Where Described	
LOOKUP_VEHICLE	Sec. 2.9.3.2 See Appendix A	
OBJ_SIMPLE_RCVQ	Sec. 2.9.3.2 See Appendix A	
buffer_simple_enqueue	Sec. 2.14.4.1.14	

Table 2.3-18: enqueue_on_rcvq CSU [8.3.2.10]

2.3.2.11 interesting_packet_test CSU

This CSU determines if message protocol and kind warrant immediate interest.

Parameters		
Parameters	Type	Where Typedef Declared
protocol	AssociationUserProtocol	p_assoc.h
pdu	pointer to char	Standard
cached_saf_id	pointer to unsigned int	Standard
ReturnValues		
Return Value	Type	Meaning
TRUE	int	Interesting
FALSE	int	Not interesting
Calls		
Function	Where Described	
saf_id from simnet id		
IS MISSILE		
LOOKUP_VEHICLE	Sec. 2.9.3.2 See Appendix A	

Table 2.3-19: interesting_packet_test CSU [8.3.2.11]

2.3.2.12 protocol_sim_process CSU

This CSU processes a simulation process if it is the correct protocol version and is not a packet with an exercise identification number attached. The processing depends on the kind of simulation PDU received and uses a switch on case.

Parameters		
Parameters	Type	Where Typedef Declared
spdu	pointer to SimulationPDU	p_sim.h
cached_saf_id	unsigned int	Standard
Calls		
Function	Where Described	
ERROR OUT	Sec. 2.5.2.2	
LOOKUP_VEHICLE	Sec. 2.9.3.2 See Appendix A	
enqueue on rcvq	Sec. 2.3.2.10	
get sender of last packet	Sec. 2.3.2.15	
simnet id string from saf id		
create remote vehicle	Sec. 2.7.1.17	
remote new appearance	Sec. 2.7.1.12	
ground input to sbx	Sec. 2.4.3.2.24	
vehicle impact to sbx	Sec. 2.4.3.2.25	
indirect fire to sbx	Sec. 2.4.3.2.23	
buffer simple enqueue	Sec. 2.14.4.1.14	
OBJ SIMPLE RCVO	Sec. 2.9.1.1 See Appendix A	

Table 2.3-20: protocol_sim_process CSU [8.3.2.12]

2.3.2.13 protocol_stealth_process CSU

This CSU first determines if the stealth protocol is of the current version, ignoring it if it is not. It then determines the kind of stealth PDU and, unless it is a stealth appearance type of PDU, ignores the PDU. Otherwise, it processes the message as required.

Parameters		
Parameters	Type	Where Typedef Declared
stlthpdu	pointer to StealthPDU	p_stlth.h
cached saf id	unsigned int	Standard
Calls		
Function	Where Described	
ERROR OUT	Sec. 2.5.2.2	
get sender of last packet	Sec. 2.3.2.15	
LOOKUP_VEHICLE	Sec. 2.9.3.2	
simnet id string from saf id		
create remote vehicle	Sec. 2.7.1.17	
remote new appearance	Sec. 2.7.1.12	

Table 2.3-21: protocol_stealth_process CSU [8.3.2.13]

2.3.2.14 protocol_data_process CSU

This CSU prints out a message questioning why the software has called this routine. It should not have received and be viewing a data packet.

Parameters		
Parameters	Type	Where Typedef Declared
dcpdu	pointer to DataCollectionDPU	p_data.h
cached saf id	unsigned int	Standard
Calls		
Function	Where Described	
ERROR OUT	Section 2.5.2.2	

Table 2.3-22: protocol_data_process CSU [8.3.2.14]

2.3.2.15 get_sender_of_last_packet CSU

This CSU returns a pointer to the network address (first six elements of buffer array) of the last message sender.

ReturnValues		
Return Value	Type	Meaning
*buf	pointer to char	Last sender address
Calls		
Function	Where Described	
AssocGetLastAddress	Sec. 2.20.1.15.1 MCC CSCI SDD	

Table 2.3-23: get_sender_of_last_packet CSU [8.3.2.15]

2.4 SAF COMMAND INTERFACE CSC

The SAF Command Interface CSC runs when ticked by the scheduler. Its code accepts and processes commands from the SAF Workstation CSCI and returns reports to it.

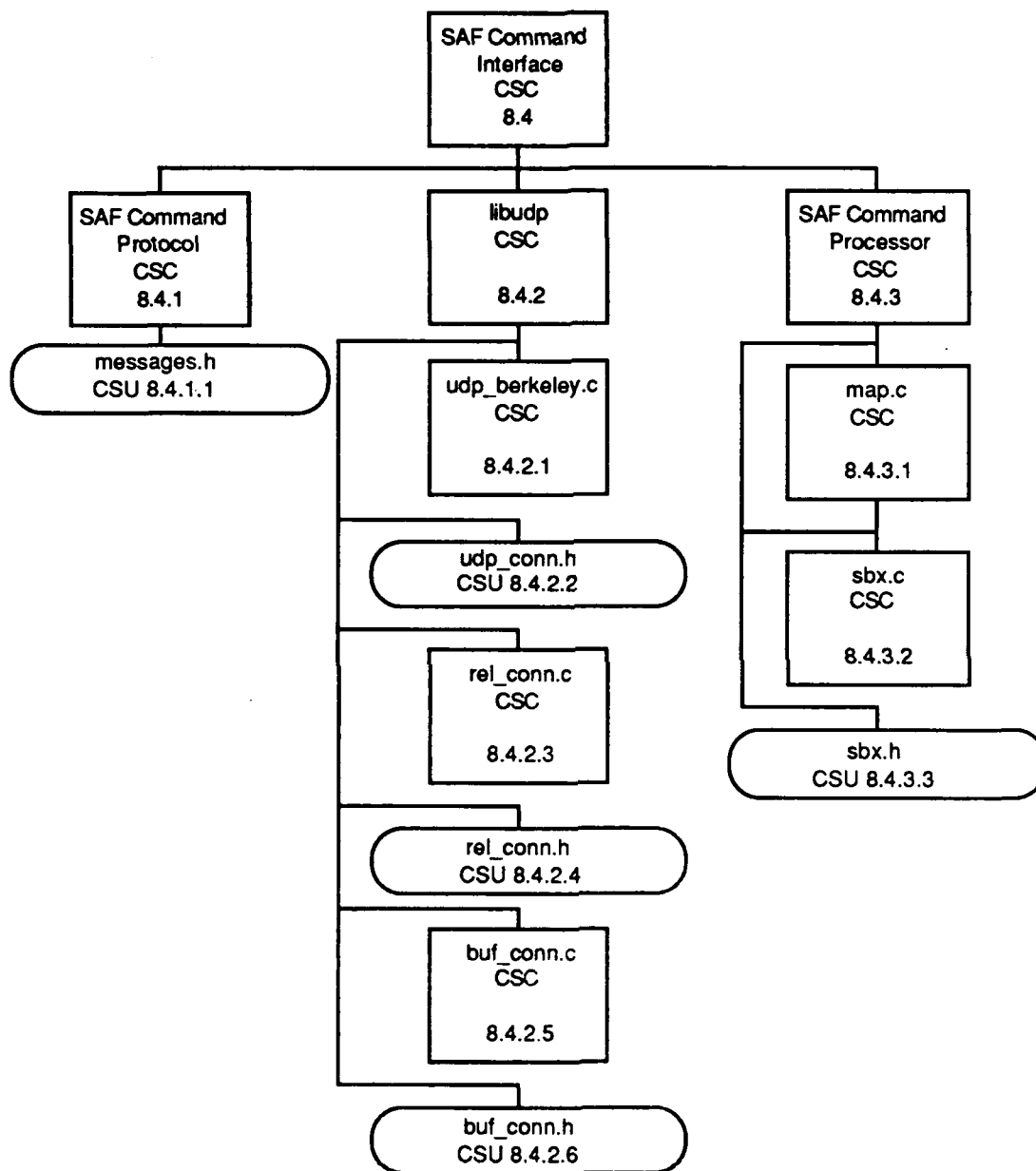


Figure 2.4-1: SAF Command Interface CSC Structure

Command messages come to the SAF Simhost HWCI via an Ethernet LAN connection. The SAF Simhost CSCI receives the commands from a UNIX socket. The command packets are IP/UDP packets. An additional network layer called RUDP is placed over the UDP layer. This RUDP layer is implemented in the SAF command interface code to provide reliable transmission of the command packets. After going through the RUDP layer, the packets are converted to SAF Command Protocol packets. When these packets

are processed and interpreted, the state of the simulation objects is changed appropriately so that when these objects next get ticked they will respond to the orders they have received from the workstation.

2.4.1 SAF Command Protocol CSC

The SAF Command Protocol CSC [8.4.1] is contained in the file messages.h.

2.4.1.1 messages.h CSU

/simnet/include/saf/src/messages.h

This file contains the message constants and formats (structures) for communication with the workstation. Each different type of information has it's own message type and constant associated with it.

The following constant definitions are the messages from Symbolics to MIPS machines.

Constant	Value
OPFOR MSG CREATE	1
OPFOR MSG RESET	2
OPFOR MSG ARTY	3
OPFOR MSG READ CONFIG	4
OPFOR MSG VEHICLE REINIT	5
OPFOR MSG RESUME	6
OPFOR MSG RESUPPLY	8
OPFOR MSG ATTACH	9
OPFOR MSG DETACH	10
OPFOR MSG TELEPORT	11
OPFOR MSG POLL	14
OPFOR MSG MINEFIELD	16
OPFOR MSG DISCONNECT	17
OPFOR MSG QUERY SUB STATE	18
OPFOR MSG IVIS XMIT MODES	19
OPFOR MSG IVIS PARAMETERS	20
OPFOR MSG CONTINUE MISSION	21
OPFOR MSG ASSIGN ROUTE	22
OPFOR MSG POINT	23
OPFOR MSG AREA	24
OPFOR MSG ZONE	25
OPFOR MSG LINE	26
OPFOR MSG ROUTE	27
OPFOR MSG DELETE OVERLAY	28
OPFOR MSG EXECUTE OVERLAY	29
OPFOR MSG HALT	30
OPFOR MSG CHANGE SPEED	31
OPFOR MSG CHANGE FORMATION	32
OPFOR MSG DELETE CM	33
OPFOR MSG FOLLOW VEHICLE	34
OPFOR MSG GOTO POINT	35
OPFOR MSG RESUME MISSION	36
OPFOR MSG FACE DIRECTION	37
OPFOR MSG SET TARGETING	38
OPFOR MSG ATTACH STEALTH	39
OPFOR MSG HOLD	40
OPFOR MSG ALTITUDE	41
OPFOR MSG ENROUTE MOVEMENT	42
OPFOR MSG SIMULATOR IN COMMAND	43
OPFOR MSG REJOIN UNIT	44
OPFOR MSG LAND	45
OPFOR MSG ATTACK	46
OPFOR MSG CHECK STATION	47

Table 2.4-1: Symbolics to MIPS Messages Constant Definitions

In the following header structure, OPFOR_HEADER, "id" is the unit id and "type" is the message type.

Item	Type	Where Type Defined
id	unsigned short	Standard
type	unsigned short	Standard

Table 2.4-2: OPFOR_HEADER Structure Definition

The generic message, OPFOR_GENERIC_MSG, consists of the previously defined header structure and the first byte of data. This structure is used to refer to messages whose type has not been defined.

Item	Type	Where Type Defined
hdr	OPFOR_HEADER	Previous structure
data	char	Standard

Table 2.4-3: OPFOR_GENERIC_MSG Structure Definition

The create message, using structure CREATE_MSG, causes a unit and all its inferior units to be instantiated by the Simhost.

Item	Type	Where Type Defined
hdr	OPFOR_HEADER	First structure in this file
forceID	unsigned char	Standard
countryD	unsigned char	Standard
countryO	unsigned char	Standard
tactics	unsigned char	Standard
echelon	unsigned char	Standard
echelon_type	unsigned char	Standard
sbx_unique_id	unsigned char	Standard
pad1	unsigned char	Standard
formation[36]	char	Standard
heading	REAL	sim_types.h
position[3]	REAL	sim_types.h
battalion	unsigned short	Standard
company	unsigned char	Standard
platoon	unsigned char	Standard
percent_amm0	float	Standard
percent_fuel	float	Standard

Table 2.4-4: CREATE_MSG Structure Definition

The reset message, whose structure is RESET_MSG, is the only message that travels in both directions between Symbolics and Simhost. A reset message from the Symbolics instructs the Simhost to delete the indicated vehicles and return the message as confirmation. A reset from the Simhost instructs the Symbolics that the indicated vehicles are no longer in the simulated world.

Item	Type	Where Type Defined
hdr	OPFOR_HEADER	First structure in this file
flags	int	Standard
unit_count	int	Standard
unit_list[32]	unsigned int	Standard

Table 2.4-5: RESET_MSG Structure Definition

If the flags field has the value RESET_ALL_VEHICLES, all vehicles created by that workstation are reset.

Constant	Value
RESET_ALL_VEHICLES	0x0001

Table 2.4-6: RESET_ALL_VEHICLES Constant Definition

The artillery message, whose structure is ARTY_MSG, instructs the Simhost to generate MCC-like artillery at the indicated position.

Item	Type	Where Type Defined
hdr	OPFOR_HEADER	First structure in this file
type	int	Standard
ammo	int	Standard
fuze	int	Standard
count	int	Standard
spread	float	Standard
position[3]	REAL	

Table 2.4-7: ARTY_MSG Structure Definition

Constant	Value
MAX_WEAPONS_ON_NET	4

Table 2.4-8: MAX_WEAPONS_ON_NET Constant Definition

The vehicle reinitialize message, whose structure is VEHICLE_REINIT_MSG, sets the state of a vehicle.

Item	Type	Where Type Defined
hdr	OPFOR_HEADER	First structure in this file
loads[MAX_WEAPONS_ON_NET]	short	Standard
fuel	int	Standard
status	char	Standard
padding[3]	char	Standard
x	float	Standard
y	float	Standard
bearing	float	Standard

Table 2.4-9: VEHICLE_REINIT_MSG Structure Definition

BBN Systems and Technologies**SAF Simulation Host CSCI**

The resupply message, whose structure is RESUPPLY_MSG, causes a vehicle to be refueled and reloaded from a resupply vehicle.

Item	Type	Where Type Defined
hdr	OPFOR HEADER	First structure in this file
resupply_vehicle	int	Standard

Table 2.4-10: RESUPPLY_MSG Structure Definition

The attach message, whose structure is ATTACH_MSG, directs an unattached unit to attach itself to a maneuver unit as an inferior vehicle. This message is unused.

Item	Type	Where Type Defined
hdr	OPFOR HEADER	First structure in this file
unit_id	int	Standard

Table 2.4-11: ATTACH_MSG Structure Definition

The detach message, whose structure is DETACH_MSG, directs a superior unit to detach a subordinate. This message is unused.

Item	Type	Where Type Defined
hdr	OPFOR HEADER	First structure in this file
unit_id	int	Standard

Table 2.4-12: DETACH_MSG Structure Definition

The teleport message, whose structure is TELEPORT_MSG, is the instantaneous movement of a vehicle so that it appears immediately at the desired location.

Item	Type	Where Type Defined
hdr	OPFOR HEADER	First structure in this file
heading	REAL	sim_types.h
position[3]	REAL	sim_types.h

Table 2.4-13: TELEPORT_MSG Structure Definition

Constant	Value
GODS EYE VIEW	0
NON GODS EYE VIEW	1
COMMANDERS EYE VIEW	2

Table 2.4-14: _VIEW Constant Definitions

The poll message, whose structure is POLL_MSG, requests vehicle data from the Simhost. The request field selects the view which determines what vehicle sends data to the Symbolics.

BBN Systems and Technologies**SAF Simulation Host CSCI**

Item	Type	Where Type Defined
hdr	OPFOR HEADER	First structure in this file
request	int	Standard

Table 2.4-15: POLL_MSG Structure Definition

The disconnect message, whose structure is DISCONNECT_MSG, tells the MIPS machine not to communicate anymore.

Item	Type	Where Type Defined
hdr	OPFOR HEADER	First structure in this file

Table 2.4-16: DISCONNECT_MSG Structure Definition

The query substate message, whose structure is QUERY_SUBSTATE_MSG, requests more information about a vehicle or composite.

Item	Type	Where Type Defined
hdr	OPFOR HEADER	First structure in this file

Table 2.4-17: QUERY_SUBSTATE_MSG Structure Definition

The IVIS transmit modes message, whose structure is IVIS_XMIT_MODES_MSG, indicates whether IVIS packets are transmitted to SIMNET and whether IVIS reports are sent to the Symbolics.

Item	Type	Where Type Defined
hdr	OPFOR HEADER	First structure in this file
xmit simnet p	int	Standard
xmit sbx p	int	Standard

Table 2.4-18: IVIS_XMIT_MODES_MSG Structure Definition

Constant	Value
SENDALL	-1
SENDNONE	0
SENDCONTACT	1
SENDSPOT	2
SENDSHELL	4

Table 2.4-19: Ivis Transmit Modes Constant Definitions

The IVIS parameters message, whose structure is IVIS_PARAMETERS_MSG, allows an operator at the workstation to change key parameters for certain IVIS reports.

Item	Type	Where Type Defined
hdr	OPFOR_HEADER	First structure in this file
cluster distance	unsigned int	Standard
decluster distance	unsigned int	Standard
spot rep range threshold	unsigned int	Standard
report monitor time msec	unsigned int	Standard
max_reappear latency msec	unsigned int	Standard

Table 2.4-20: IVIS_PARAMETERS_MSG Structure Definition

The following typedef struct is tagged float_xy_point.

Item	Type	Where Type Defined
x	float	Standard
y	float	Standard

Table 2.4-21: FLOAT_XY_POINT Structure Definition

While x and y point coordinates are received as float type, they are stored internally as REALs. The following typedef struct is tagged xy_point.

Item	Type	Where Type Defined
x	REAL	sim_types.h
y	REAL	sim_types.h

Table 2.4-22: XY_POINT Structure Definition

The following typedef struct is tagged sbx_route_pt. A route point is typed by a control measure id. If the id is even, x and y are coordinate points on the terrain. If id is odd, x is a road segment id and y is a direction.

Item	Type	Where Type Defined
id	int	Standard
x	int	Standard
y	int	Standard

Table 2.4-23: SBX_ROUTE_PT Structure Definition

The following ASSIGN_ROUTE_MSG structure is unused.

Item	Type	Where Type Defined
hdr	OPFOR_HEADER	First structure in this file
count	int	Standard
waypoints[20]	FLOAT_XY_POINT	Earlier structure in this file

Table 2.4-24: ASSIGN_ROUTE_MSG Structure Definition

Constant	Value
CM NAME SIZE	20
CM TYPE SIZE	20
CIS NAME SIZE	40

Table 2.4-25: Control Measure Constant Definitions

The following typedef struct is tagged `cm_id`. It identifies a control measure by a unique integer assigned by the workstation.

Item	Type	Where Type Defined
<code>name[CM NAME SIZE]</code>	char	Standard
<code>id</code>	int	Standard

Table 2.4-26: CM_ID Structure Definition

The following typedef struct is tagged `float_cm_point_list`. Like points, point lists are received as float type, but are internally stored as REAL.

Item	Type	Where Type Defined
<code>count</code>	int	Standard
<code>waypoints[20]</code>	FLOAT_XY_POINT	Earlier structure in this file

Table 2.4-27: FLOAT_CM_POINT_LIST Structure Definition

The following typedef struct is tagged `cm_point_list`.

Item	Type	Where Type Defined
<code>count</code>	int	Standard
<code>waypoints[20]</code>	XY_POINT	Earlier structure in this File

Table 2.4-28: CM_POINT_LIST Structure Definition

Constant	Value
NUM_APPLICABLE_UNITS	32

Table 2.4-29: NUM_APPLICABLE_UNITS Constant Definition

The following typedef struct is tagged point_msg. It creates a new point control measure as identified by its point id.

Item	Type	Where Type Defined
hdr	OPFOR HEADER	First structure in this file
overlay	CM ID	Earlier structure in this file
point	CM ID	Earlier structure in this file
x	float	Standard
y	float	Standard
route	CM ID	Earlier structure in this file
speed	float	Standard
cis[CIS_NAME_SIZE]	char	Standard
report	short	Standard
num units	short	Standard
applies to[NUM APPLICABLE UNITS]	short	Standard

Table 2.4-30: POINT_MSG Structure Definition

The following typedef struct is tagged area_msg. It creates a new area control measure as identified by its area id.

Item	Type	Where Type Defined
hdr	OPFOR HEADER	First structure in this file
overlay	CM ID	Earlier structure in this file
area	CM ID	Earlier structure in this file
points	FLOAT CM POINT LIST	Earlier structure in this file
type[CM_TYPE_SIZE]	char	Standard
speed	float	Standard
cis[CIS_NAME_SIZE]	char	Standard
report	short	Standard
num units	short	Standard
applies to[NUM APPLICABLE UNITS]	short	Standard

Table 2.4-31: AREA_MSG Structure Definition

The following typedef struct is tagged zone_msg. It creates a new zone control measure as identified by its zone id.

Item	Type	Where Type Defined
hdr	OPFOR HEADER	First structure in this file
overlay	CM ID	Earlier structure in this file
zone	CM ID	Earlier structure in this file
points	FLOAT CM POINT LIST	Earlier structure in this file
type[CM_TYPE_SIZE]	char	Standard
speed	float	Standard
cis[CIS_NAME_SIZE]	char	Standard
report	short	Standard
num units	short	Standard
applies to[NUM APPLICABLE UNITS]	short	Standard

Table 2.4-32: ZONE_MSG Structure Definition

The following typedef struct is tagged line_msg. It creates a new line control measure as identified by its line id.

Item	Type	Where Type Defined
hdr	OPFOR_HEADER	First structure in this file
overlay	CM_ID	Earlier structure in this file
line	CM_ID	Earlier structure in this file
points	FLOAT_CM_POINT_LIST	Earlier structure in this file
type[CM_TYPE_SIZE]	char	Standard
speed	float	Standard
cis[CIS_NAME_SIZE]	char	Standard
report	short	Standard
num_units	short	Standard
applies_to[NUM_APPLICABLE_UNITS]	short	Standard

Table 2.4-33: LINE_MSG Structure Definition

The following typedef struct is tagged route_msg. It creates a new route control measure as identified by its route id.

Item	Type	Where Type Defined
hdr	OPFOR_HEADER	First structure in this file
overlay	CM_ID	Earlier structure in this file
route	CM_ID	Earlier structure in this file
count	int	Standard
points[100]	SBX_ROUTE_POINT	Earlier structure in this file
num_units	short	Standard
applies_to[NUM_APPLICABLE_UNITS]	short	Standard

Table 2.4-34: ROUTE_MSG Structure Definition

The following typedef struct is tagged delete_overlay_msg. This message causes all control measures within an overlay to be removed.

Item	Type	Where Type Defined
hdr	OPFOR_HEADER	First structure in this file
overlay	CM_ID	Earlier structure in this file

Table 2.4-35: DELETE_OVERLAY_MSG Structure Definition

The following typedef struct is tagged execute_overlay_msg. This message causes a unit to follow the indicated route and to react to the control measures that exist in the overlay.

Item	Type	Where Type Defined
hdr	OPFOR_HEADER	First structure in this file
overlay	CM_ID	Earlier structure in this file
route	CM_ID	Earlier structure in this file
initial_cis[CIS_NAME_SIZE]	char	Standard

Table 2.4-36: EXECUTE_OVERLAY_MSG Structure Definition

The following typedef struct is tagged `delete_cm_msg`. This message removes a control measure from the overlay.

Item	Type	Where Type Defined
hdr	OPFOR HEADER	First structure in this file
overlay	CM_ID	Earlier structure in this file
counter_measure	CM_ID	Earlier structure in this file

Table 2.4-37: DELETE_CM_MSG Structure Definition

The following typedef struct is tagged `rejoin_unit`. This message causes a unit to abandon its current mission, as assigned by an execute overlay message, and to rejoin the mission of its superior unit, if a superior unit exists.

Item	Type	Where Type Defined
hdr	OPFOR HEADER	First structure in this file

Table 2.4-38: REJOIN_UNIT_MSG Structure Definition

The following messages cause immediate intervention that supercedes missions assigned in overlays. The following typedef struct is tagged `halt_msg`. It causes a unit to stop movement.

Item	Type	Where Type Defined
hdr	OPFOR HEADER	First structure in this file

Table 2.4-39: HALT_MSG Structure Definition

The following typedef struct is tagged `resume_msg`. This message causes a unit to resume its pre-planned mission after the unit has been interrupted by one of the immediate interventions.

Item	Type	Where Type Defined
hdr	OPFOR HEADER	First structure in this file

Table 2.4-40: RESUME_MSG Structure Definition

The following typedef struct is tagged `change_speed_msg`. It causes a unit to change speed.

Item	Type	Where Type Defined
hdr	OPFOR HEADER	First structure in this file
speed	int	Standard

Table 2.4-41: CHANGE_SPEED_MSG Structure Definition

The following typedef struct is tagged change_formation_msg. It causes a unit to change formation.

Item	Type	Where Type Defined
hdr	OPFOR_HEADER	First structure in this file
formation[20]	char	Standard

Table 2.4-42: CHANGE_FORMATION_MSG Structure Definition

The following typedef struct is tagged follow_vehicle_msg. It causes a unit to start following a specified vehicle.

Item	Type	Where Type Defined
hdr	OPFOR_HEADER	First structure in this file
xoff /* In leader's coordinate */	float	Standard
yoff /* system */	float	Standard
leadid	unsigned short	Standard
padding	unsigned short	Standard

Table 2.4-43: FOLLOW_VEHICLE_MSG Structure Definition

The following typedef struct is tagged goto_point_msg. It causes a unit to go to a specified location.

Item	Type	Where Type Defined
hdr	OPFOR_HEADER	First structure in this file
x	float	Standard
y	float	Standard
backwardp	int	Standard
type	int	Standard

Table 2.4-44: GOTO_POINT_MSG Structure Definition

The following typedef struct is tagged resume_mission_msg. This message functions in the same manner as resume_msg.

Item	Type	Where Type Defined
hdr	OPFOR_HEADER	First structure in this file

Table 2.4-45: RESUME_MISSION_MSG Structure Definition

The following typedef struct is tagged face_direction_msg. This causes a unit to face in a specified direction.

Item	Type	Where Type Defined
hdr	OPFOR_HEADER	First structure in this file
mathradians	float	Standard

Table 2.4-46: FACE_DIRECTION_MSG Structure Definition

Constant	Value
HOVER HOLD	0
ORBIT HOLD	1
RACETRACK HOLD	2
LAND HOLD	3

Table 2.4-47: _HOLD Constant Definitions

The following typedef struct is tagged hold_msg. This causes an air-unit to stop at its current location and either hover, orbit, racetrack or land at that location.

Item	Type	Where Type Defined
hdr	OPFOR_HEADER	First structure in this file
type	int	Standard

Table 2.4-48: HOLD_MSG Structure Definition

Constant	Value
ABS ALTITUDE	0
REL ALTITUDE	1

Table 2.4-49: _ALTITUDE Constant Definitions

The following typedef struct is tagged altitude_msg. This causes an air-unit to change its altitude.

Item	Type	Where Type Defined
hdr	OPFOR_HEADER	First structure in this file
altitude	float	Standard
type	int	Standard

Table 2.4-50: ALTITUDE_MSG Structure Definition

Constant	Value
TAC COL	0
BOUND	1
BOUND_OVER	2

Table 2.4-51: Enroute Movement Constant Definitions

The following typedef struct is tagged enroute_movement. This message is unused.

Item	Type	Where Type Defined
hdr	OPFOR_HEADER	First structure in this file
type	int	Standard

Table 2.4-52: ENROUTE_MOVEMENT_MSG Structure Definition

The following typedef struct is tagged `simulator_in_command`. This message causes the lead vehicle of a unit to be deactivated. The unit is thereafter lead by the simulator indicated in the message.

Item	Type	Where Type Defined
hdr	OPFOR HEADER	First structure in this file
leadid	unsigned short	Standard
padding	unsigned short	Standard

Table 2.4-53: SIMULATOR_IN_COMMAND_MSG Structure Definition

The following typedef struct is tagged `land`. This message causes an air-unit to land at the specified location.

Item	Type	Where Type Defined
hdr	OPFOR HEADER	First structure in this file
x	float	Standard
y	float	Standard

Table 2.4-54: LAND_MSG Structure Definition

Constant	Value
ATTACK_RUNNING_FIRE	1
ATTACK_HOVER_FIRE	2

Table 2.4-55: attack_type Constant Definitions

The following typedef struct is tagged `attack`. This message causes the air-unit to attack between the start and target locations. The type of attack is specified as an `attack_type`.

Item	Type	Where Type Defined
hdr	OPFOR HEADER	First structure in this file
attack_type	int	Standard
start_x	float	Standard
start_y	float	Standard
target_x	float	Standard
target_y	float	Standard

Table 2.4-56: ATTACK_MSG Structure Definition

The following typedef struct is tagged check_station. This message is unused.

Item	Type	Where Type Defined
hdr	OPFOR_HEADER	First structure in this file

Table 2.4-57: CHECK_STATION_MSG Structure Definition

The following typedef struct is tagged set_targeting_msg. This message causes a vehicle or unit to change its targeting parameters.

Item	Type	Where Type Defined
hdr	OPFOR_HEADER	First structure in this file
firestatus	int	Standard
max_engagement_range	int	Standard
marksmanship	float	Standard
position_x	float	Standard
position_y	float	Standard
radius	float	Standard
targets[16]	unsigned short	Standard

Table 2.4-58: SET_TARGETING_MSG Structure Definition

The following typedef struct is tagged attach_stealth_msg. This message causes the stealth vehicle indicated by the site and host to attach in mimic mode to the indicated vehicle in the OPFOR_HEADER.

Item	Type	Where Type Defined
hdr	OPFOR_HEADER	First structure in this file
site	short	Standard
host	short	Standard

Table 2.4-59: ATTACH_STEALTH_MSG Structure Definition

The following table contains the Simhost to Symbolics message constant definitions.

Constant	Value
OPFOR MSG CREATION	100
OPFOR MSG WHERE ARE THEY	101
OPFOR MSG GROUND IMPACT	102
OPFOR MSG VEHICLE IMPACT	103
OPFOR MSG INTERVISIBILITY	104
OPFOR MSG NOTIFY	105
OPFOR MSG VEHICLE STATUS	106
OPFOR MSG INDIRECT FIRE	108
OPFOR MSG ACTIVITY COMPLETE	111
OPFOR MSG MACHINE STATUS	114
OPFOR MSG MINEFIELD CREATION	115
OPFOR MSG SUB STATE	116
OPFOR MSG IVIS CONTACT	117
OPFOR MSG IVIS SPOT	118
OPFOR MSG IVIS SHELL	119
OPFOR MSG VEHICLE PAE	120
OPFOR MSG VEHICLE POSITION	121
OPFOR MSG VEHICLE POSITON POLL COMPLETED	122
OPFOR MSG VEHICLE APPEARANCE	123
OPFOR MSG VEHICLE ECHELON	124
OPFOR MSG GENERIC MESSAGE	125
OPFOR MSG STEALTH POSITION	126
OPFOR MSG VEHICLE LOAD	127

Table 2.4-60: Simhost to Symbolics Messages Constant Definitions

GROUND_IMPACT_MSG informs the Symbolics of a ground-impact on the battlefield.

Item	Type	Where Type Defined
hdr	OPFOR_HEADER	First structure in this file
ammunition	unsigned char	Standard
quantity	unsigned char	Standard
padding	short	Standard
locx	float	Standard
locy	float	Standard

Table 2.4-61: GROUND_IMPACT_MSG Structure Definition

VEHICLE_IMPACT_MSG informs the Symbolics of a vehicle-impact on the battlefield.

Item	Type	Where Type Defined
hdr /* Contains attacker id */	OPFOR_HEADER	First structure in this file
target id	int	Standard
round type	int	Standard
burst length	int	Standard

Table 2.4-62: VEHICLE_IMPACT_MSG Structure Definition

MACHINE_STATUS_MSG is sent to the Symbolics when a connection is established.

Item	Type	Where Type Defined
hdr	OPFOR HEADER	First structure in this file
real time clock value	unsigned int	Standard

Table 2.4-63: MACHINE_STATUS_MSG Structure Definition

INDIRECT_FIRE_MSG informs the Symbolics of indirect fire on the battlefield. It contains one or more indirect-fire bursts.

Item	Type	Where Type Defined
hdr	OPFOR HEADER	First structure in this file
ammunition	unsigned char	Standard
fuze	unsigned char	Standard
quantity	unsigned char	Standard
rate	unsigned char	Standard

Table 2.4-64: INDIRECT_FIRE_MSG Structure Definition

Item	Type	Where Type Defined
locx	float	Standard
locy	float	Standard
locz	float	Standard
delay	short	Standard
padding	short	Standard

Table 2.4-65: INDIRECT_FIRE_BURST Structure Definition

The following typedef struct is tagged vehicle_postion_descriptor. It describes the position, direction and turret angle of a vehicle.

Item	Type	Where Type Defined
x /* Simnet x position */	float	Standard
y /* Simnet y position */	float	Standard
direction angle	float	Standard
turret_angle	float	Standard

Table 2.4-66: VEHICLE_POSITION_DESCRIPTOR Structure Definition

The following typedef struct is tagged `vehicle_appearance_descriptor`. It describes the appearance of a vehicle.

Item	Type	Where Type Defined
icon /* Appearance */	unsigned char	Standard
status /* SAF status data */	unsigned char	Standard
force /* Simnet force data */	unsigned short	Standard
tactics /* SAF tactics data */	unsigned char	Standard
marking[11] /* Marking field */	char	Standard

Table 2.4-67: VEHICLE_APPEARANCE_DESCRIPTOR Structure Definition

The following typedef struct is tagged `vehicle_echelon_descriptor`. It describes the position of a unit or vehicle within a unit hierarchy.

Item	Type	Where Type Defined
port number	int	Standard
echelon	unsigned char	Standard
job_desc	unsigned char	Standard
superior_id	unsigned short	Standard
sbx_uniq_id	unsigned char	Standard
top_supervisor_uniq_id	unsigned char	Standard
relative_id	unsigned char	Standard
inf_count	unsigned char	Standard
inferiors[8]	unsigned short	Standard

Table 2.4-68: VEHICLE_ECHELON_DESCRIPTOR Structure Definition

The structure for the position, appearance, and echelon message is tagged `vehicle_pae_msg`. It contains all the information about a vehicle and only needs to be sent at creation time.

Item	Type	Where Type Defined
hdr	OPFOR HEADER	First structure in this file
position	VEHICLE POSITION DESCRIPTOR	Earlier structure in this file
appearance	VEHICLE APPEARANCE DESCRIPTOR	Earlier structure in this file
echelon	VEHICLE ECHELO' DESCRIPTOR	Earlier structure in this file

Table 2.4-69: VEHICLE_PAE_MSG Structure Definition

The following typedef struct is tagged `vehicle_position_msg`. The vehicle position message is sent to update a vehicle's current position.

Item	Type	Where Type Defined
hdr	OPFOR HEADER	First structure in this file
position	VEHICLE POSITION DESCRIPTOR	Earlier structure in this file

Table 2.4-70: VEHICLE_POSITION_MSG Structure Definition

The following typedef struct is tagged `vehicle_load_msg`. The vehicle load message is required to check a vehicle's ammunition and fuel loads.

Item	Type	Where Type Defined
hdr	OPFOR HEADER	First structure in this file
loads[MAX WEAPONS ON NET]	short	Standard
fuel	int	Standard

Table 2.4-71: VEHICLE_LOAD_MSG Structure Definition

The following typedef struct is tagged `vehicle_position_poll_completed`. This message is needed to indicate the completion of the transmitting of all vehicle position messages for the current round.

Item	Type	Where Type Defined
hdr	OPFOR HEADER	First structure in this file

Table 2.4-72: VEHICLE_POSITION_POLL_COMPLETED_MSG Structure Definition

The following typedef struct is tagged `vehicle_appearance_msg`. The vehicle appearance message is needed to update a vehicle's appearance when it gets damaged.

Item	Type	Where Type Defined
hdr	OPFOR HEADER	First structure in this file
appearance	VEHICLE APPEARANCE DESCRIPTOR	Earlier structure in this file

Table 2.4-73: VEHICLE_APPEARANCE_MSG Structure Definition

The following typedef struct is tagged `vehicle_echelon_msg`. The vehicle echelon message is sent if the position within the unit hierarchy changes.

Item	Type	Where Type Defined
hdr	OPFOR HEADER	First structure in this file
echelon	VEHICLE ECHELON DESCRIPTOR	Earlier structure in this file

Table 2.4-74: VEHICLE_ECHELON_MSG Structure Definition

The following constants and structures are used to print an arbitrary message at the Symbolics.

Constant	Value
GENERIC MESSAGE MAX	1024

Table 2.4-75: Maximum Characters Constant Definition

Constant	Value
GM RADIO MESSAGE	0
GM ERROR MESSAGE	1
GM RADIO ALERT MESSAGE	2

Table 2.4-76: Generic Message Type Constant Definitions

The following typedef struct is tagged generic_message_msg.

Item	Type	Where Type Defined
hdr	OPFOR HEADER	First structure in this file
type	int	Standard
message[GENERIC_MESSAGE_MAX]	char	Standard

Table 2.4-77: GENERIC_MESSAGE_MSG Structure Definition

The following typedef struct is tagged stealth_position_msg. It is used to inform the Symbolics of the location of a stealth.

Item	Type	Where Type Defined
hdr	OPFOR HEADER	First structure in this file
position	VEHICLE_POSITION_DESCRIPTOR	Earlier structure in this file

Table 2.4-78: STEALTH_POSITION_MSG Structure Definition

2.4.2 libudp CSC

/simnet/libsrc/libudp

This library is used by the Simhost to communicate with the workstation, specifically to send and receive the packets. It uses the standard UDP protocol.

2.4.2.1 udp_berkeley.c CSC

/simnet/libsrc/libudp/udp_berkeley.c

The udp_berkeley.c CSC contains eleven CSUs dealing with the UDP network layer. These CSUs are displayed in Figure 2.4-2.

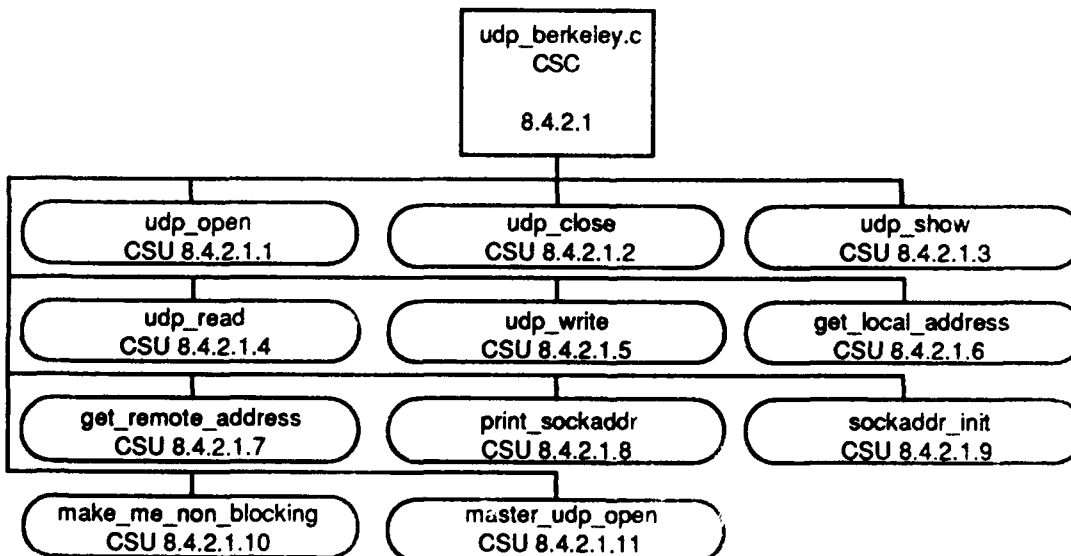


Figure 2.4-2: udp_berkeley.c CSC Structure

2.4.2.1.1 udp_open CSU

This CSU gets the necessary connection memory block, creates the UDP socket, makes the interface non-blocking, and binds it to the local address accompanied by a unique port number which is assigned to a Symbolic workstation. This procedure ensures that only the Symbolics can communicate with this connection.

Parameters		
Parameters	Type	Where Typedef Declared
local_addr	pointer to char	Standard
local_port	int	Standard
ReturnValues		
Return Value	Type	Meaning
(u_ptr) = NULL	pointer to UDP_CONNECTION	No connection block
(u_ptr) <> NULL	pointer to UDP_CONNECTION	Pointer to connection block
Calls		
Function	Where Described	
heap_calloc	Sec. 2.14.2.1.2	
gethostname	Sec. 2.5.1.10	
sockaddr_init	Sec. 2.4.2.1.9	
socket		
make_me_non_blocking	Sec. 2.4.2.1.10	

Table 2.4-79: udp_open CSU [8.4.2.1.1]

2.4.2.1.2 udp_close CSU

This CSU closes the UDP connection and releases the connection memory block.

Parameters		
Parameters	Type	Where Typedef Declared
u_ptr	pointer to UDP_CONNECTION	Sec. 2.4.2.2
Calls		
Function	Where Described	
heap_deallocate	Sec. 2.14.2.1.4	

Table 2.4-80: udp_close CSU [8.4.2.1.2]

2.4.2.1.3 udp_show CSU

This CSU prints the UDP connection including the socket and both local and remote hosts.

Parameters		
Parameters	Type	Where Typedef Declared
u_ptr	pointer to UDP_CONNECTION	Sec. 2.4.2.2
Calls		
Function	Where Described	
print_sockaddr	Sec. 2.4.2.1.8	

Table 2.4-81: udp_show CSU [8.4.2.1.3]

2.4.2.1.4 udp_read CSU

This CSU attempts to read the UDP connection to the remote port.

Parameters		
Parameters	Type	Where Typedef Declared
u_ptr	pointer to UDP_CONNECTION	Sec. 2.4.2.2
buffer_ptr	pointer to char	Standard
max_len	int	Standard
ReturnValues		
ReturnValue	Type	Meaning
0	int	Socket read error or handshaking
(bytes read)	int	No. of data bytes
Errors		
Error Name	Reason for Error	
byte_read == -1	Error in UDP socket read	
Calls		
Function	Where Described	
recvfrom		
print_sockaddr	Sec. 2.4.2.1.8	
udp_write	Sec. 2.4.2.1.5	

Table 2.4-82: udp_read CSU [8.4.2.1.4]

2.4.2.1.5 udp_write CSU

This CSU writes to the sbx, printing messages if there is a failure to write to the foreign port over the UDP, if there is a zero byte length sent, or if there is a short write to the sbx.

Parameters		
Parameters	Type	Where Typedef Declared
u_ptr	pointer to UDP_CONNECTION	Sec. 2.4.2.2
buffer_ptr	pointer to char	Standard
buf_len	int	Standard
Calls		
Function	Where Described	
sendto		
print_sockaddr	Sec. 2.4.2.1.8	

Table 2.4-83: udp_write CSU [8.4.2.1.5]

2.4.2.1.6 get_local_address CSU

This CSU gets the address of the local host.

ReturnValues		
Return Value	Type	Meaning
(inet_ntoa(...))	pointer to char	Local host address
Calls		
Function	Where Described	
gethostname	2.5.1.10	
gethostbyname	2.5.1.10	
inet_ntoa		

Table 2.4-84: get_local_address CSU [8.4.2.1.6]

2.4.2.1.7 get_remote_address CSU

This CSU, given the host name as a string, returns the internet address in the string form A.B.C.D.

Parameters		
Parameters	Type	Where Typedef Declared
hostname	pointer to char	Standard
ReturnValues		
Return Value	Type	Meaning
NULL	pointer to char	Can't get host name
(inet_ntoa(...))	pointer to char	Internet address

Calls	
Function	Where Described
gethostbyname	2.5.1.10
inet_ntoa	

Table 2.4-85: get_remote_address CSU [8.4.2.1.7]

2.4.2.1.8 print_sockaddr CSU

This CSU prints the socket address according to the machine in use.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to char	Standard
sktp	pointer to struct sockaddr_in	Sec. 2.4.2.2

Table 2.4-86: print_sockaddr CSU [8.4.2.1.8]

2.4.2.1.9 sockaddr_init CSU

This CSU initializes the contents of the socket.

Parameters		
Parameters	Type	Where Typedef Declared
sktp	pointer to struct sockaddr_in	Sec. 2.4.2.2
ascii_addr	pointer to char	Standard
port	int	Standard
Calls		
Function	Where Described	
inet_addr		
htons		

Table 2.4-87: sockaddr_init CSU [8.4.2.1.9]

2.4.2.1.10 make_me_non_blocking CSU

This CSU attempts to make the mode of the udp connection non-blocking.

Parameters		
Parameters	Type	Where Typedef Declared
skt	int	Standard
ReturnValues		
Return Value	Type	Meaning
-1	int	Non-blocking mode failed
0	int	Non-blocking mode successful

Calls	
Function	Where Described
ioctl	

Table 2.4-88: make_me_non_blocking CSU [8.4.2.1.10]

2.4.2.1.11 master_udp_open CSU

This CSU acquires the remote and the local addresses, and establishes a socket. It then calls sockaddr_init to fill in the remote address and tries to establish handshaking between the two ends. This CSU is unused.

Parameters		
Parameters	Type	Where Typedef Declared
host	pointer to char	Standard
port	int	Standard
ReturnValues		
Return Value	Type	Meaning
NULL	pointer to UDP_CONNECTION	Remote host is unknown
u_ptr=sockaddr	pointer to UDP_CONNECTION	Connection successful
u_ptr=NULL	pointer to UDP_CONNECTION	Connection failed
Calls		
Function	Where Described	
get remote address	Sec. 2.4.2.1.7	
udp_open	Sec. 2.4.2.1.1	
get local address	Sec. 2.4.2.1.6	
sockaddr_init	Sec. 2.4.2.1.9	
print sockaddr	Sec. 2.4.2.1.8	
udp_write	Sec. 2.4.2.1.5	
udp_read	Sec. 2.4.2.1.4	

Table 2.4-89: master_udp_open CSU [8.4.2.1.11]

2.4.2.2 udp_conn.h CSU

/simnet/libsrc/libudp/udp_conn.h

This CSU is the UDP layer connection include file. The contents are conditional on UDP_CONN_ALREADY_INCLUDE not being defined.

It begins by defining SOCKADDR_IN as being a typedef struct sockaddr_in. Two of these structures (one foreign and one local) are required for this CSU's one structure definition, UDP_CONNECTION, which follows.

Item	Type	Where Type Defined
udp_skt	int	Standard
foreign	SOCKADDR IN	See above
local	SOCKADDR IN	See above

Table 2.4-90: UDP_CONNECTION Structure Definition

In addition to the structure definition, the CSU contains two constant definitions.

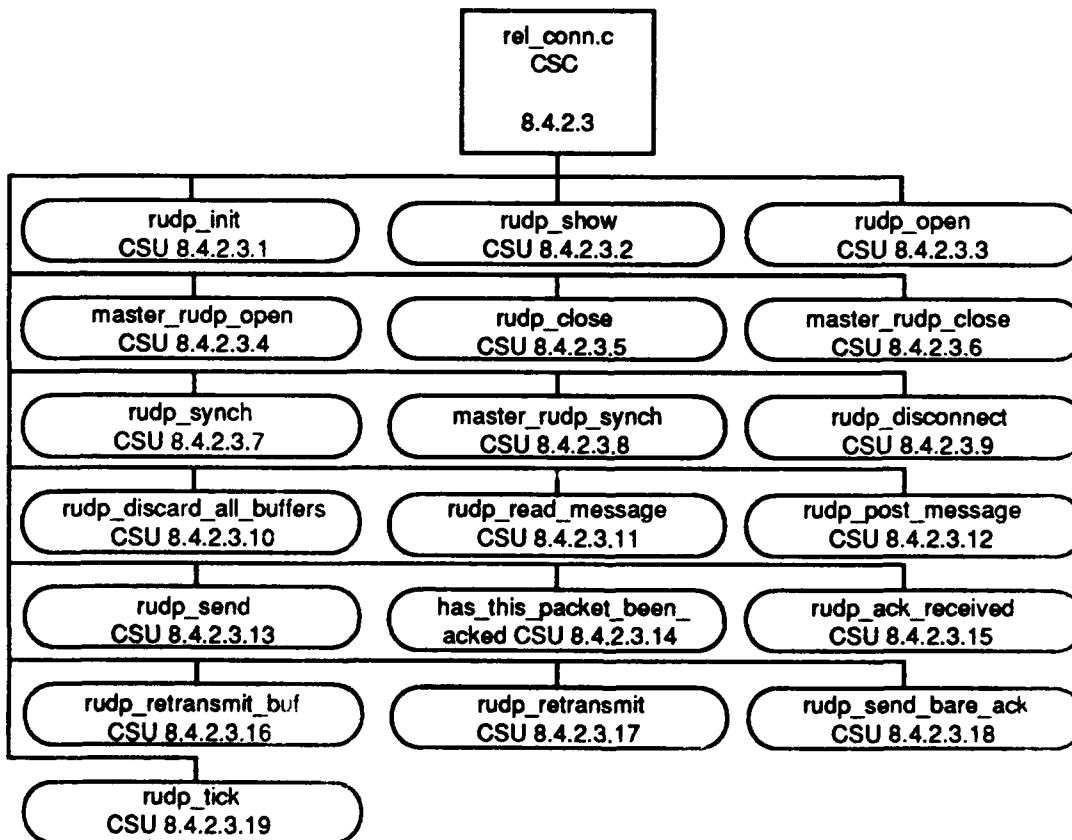
Constant	Value
BUFLN	1471 /* Max ethernet packet length */
DATALN	1400 /* Max minus room for (header + extra) */

Table 2.4-91: udp_conn.h Constant Definitions

2.4.2.3 rel_conn.c CSC

/simnet/libsrc/libudp/rel_conn.c

The rel_conn.c CSC contains nineteen CSUs (functions) dealing with the RUDP network layer connection. These CSUs are shown in Figure 2.4-3.

**Figure 2.4-3: rel_conn.c CSC Structure**

2.4.2.3.1 rudp_init CSU

This CSU initializes the RUDP socket structure pointed to by the input parameter.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to RUDP_SOCKET	Sec. 2.4.2.4
Calls		
Function	Where Described	
get_millisecond_time	Sec. 2.14.3.3.2	
heap_allocate	Sec. 2.14.2.1.4	
queue_allocate	Sec. 2.14.4.2.1	

Table 2.4-92: rudp_init CSU [8.4.2.3.1]

2.4.2.3.2 rudp_show CSU

This CSU prints the statistics for the RUDP socket passed in the parameter.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to RUDP_SOCKET	Sec. 2.4.2.4
Calls		
Function	Where Described	
get_millisecond_time	Sec. 2.14.3.3.2	
udp_show	Sec. 2.4.2.1.3	

Table 2.4-93: rudp_show CSU [8.4.2.3.2]

2.4.2.3.3 rudp_open CSU

This CSU opens a UDP connection, gets the time, and updates the RUDP socket. If BENCHMARK is defined, it also calls the sockaddr_init CSU and bcopy.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to RUDP_SOCKET	Sec. 2.4.2.4
local_addr	pointer to char	Standard
local_port	int	Standard
Calls		
Function	Where Described	
udp_open	Sec. 2.4.2.1.1	
get_millisecond_time	Sec. 2.14.3.3.2	
sockaddr_init	Sec. 2.4.2.1.9	

Table 2.4-94: rudp_open CSU [8.4.2.3.3]

2.4.2.3.4 master_rudp_open CSU

This CSU attempts to establish a UDP connection and, if successful, updates the RUDP socket structure and synchronizes the two sides of the connection. This CSU is unused.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to RUDP_SOCKET	Sec. 2.4.2.4
host	pointer to char	Standard
port	int	Standard
Return Values		
Return Value	Type	Meaning
-1	int	No connection
Calls		
Function	Where Described	
master_udp_open	Sec. 2.4.2.1.11	
get_millisecond_time	Sec. 2.14.3.3.2	
master_rudp_synch	Sec. 2.4.2.3.8	

Table 2.4-95: master_rudp_open CSU [8.4.2.3.4]

2.4.2.3.5 rudp_close CSU

This CSU closes a connection through a call to udp_close.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to RUDP_SOCKET	Sec. 2.4.2.4
Calls		
Function	Where Described	
udp_close	Sec. 2.4.2.1.2	

Table 2.4-96: rudp_close CSU [8.4.2.3.5]

2.4.2.3.6 master_rudp_close CSU

This CSU calls rudp_close. This CSU is unused.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to RUDP_SOCKET	Sec. 2.4.2.4
Calls		
Function	Where Described	
rudp_close	Sec. 2.4.2.3.5	

Table 2.4-97: master_rudp_close CSU [8.4.2.3.6]

2.4.2.3.7 rudp_synch CSU

This CSU updates the RUDP connection state, if necessary, and synchronizes the connection. It deletes all previous data and resets the packet sequence numbers.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to RUDP_SOCKET	Sec. 2.4.2.4
in_seq	unsigned int	Standard
out_seq	unsigned int	Standard
Calls		
Function	Where Described	
rudp_discard_all_buffers	Sec. 2.4.2.3.10	
DEBUG_CONNECTION	Sec. 2.4.2.4 See Appendix A	

Table 2.4-98: rudp_synch CSU [8.4.2.3.7]

2.4.2.3.8 master_rudp_synch CSU

This CSU synchronizes both ends of the connection, setting the local end through a call to CSU rudp_synch and then the slave end by allocating and enqueueing a message buffer and calling rudp_send. The message buffer is then released. This CSU is unused.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to RUDP_SOCKET	Sec. 2.4.2.4
Calls		
Function	Where Described	
rudp_synch	Sec. 2.4.2.3.7	
buffer_allocate	Sec. 2.14.4.2.12	
buffer_enqueue	Sec. 2.14.4.2.18	
rudp_send	Sec. 2.4.2.3.13	
buffer_deallocate	Sec. 2.14.4.2.15	

Table 2.4-99: master_rudp_synch CSU [8.4.2.3.8]

2.4.2.3.9 rudp_disconnect CSU

This CSU, if the RUDP socket connect state is not already closed, changes the state to "closed" and discards all socket buffers.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to RUDP_SOCKET	Sec. 2.4.2.4

Calls	
Function	Where Described
rudp_discard_all_buffers	Sec. 2.4.2.3.10
DEBUG CONNECTION	Sec. 2.4.2.4 See Appendix A

Table 2.4-100: rudp_disconnect CSU [8.4.2.3.9]

2.4.2.3.10 rudp_discard_all_buffers CSU

This CSU removes the retransmit queue for the RUDP socket passed as a parameter.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to RUDP_SOCKET	Sec. 2.4.2.4
Calls		
Function	Where Described	
buffer flush	Sec. 2.14.4.2.23	
DEBUG CONNECTION	Sec. 2.4.2.4 See Appendix A	

Table 2.4-101: rudp_discard_all_buffers CSU [8.4.2.3.10]

2.4.2.3.11 rudp_read_message CSU

This CSU attempts to read a message according to the RUDP protocol. Acknowledgements for previously sent messages are processed, and out of sequence packets are ignored.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to RUDP_SOCKET	Sec. 2.4.2.4
ReturnValues		
Return Value	Type	Meaning
0	int	Nothing to read
-1	int	Out of sequence or bare ack
number of bytes read	int	Number of message bytes
Calls		
Function	Where Described	
udp_read	Sec. 2.4.2.1.4	
DEBUG CONNECTION	Sec. 2.4.2.4 See Appendix A	
rudp_synch	Sec. 2.4.2.3.7	
rudp_ack_received	Sec. 2.4.2.3.15	

Table 2.4-102: rudp_read_message CSU [8.4.2.3.11]

2.4.2.3.12 rudp_post_message CSU

This CSU enqueues a packet to be sent, unless the retransmission queue has grown too large. The message is wrapped with an RUDP header and rudp_send is called to send out the message. The message is enqueued on the retransmission queue in case the message must be retransmitted. An acknowledgement number for the last packet received is included in the RUDP header.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to RUDP_SOCKET	Sec. 2.4.2.4
data	pointer to char	Standard
data len	int	Standard
ReturnValues		
Return Value	Type	Meaning
POST_QUEUE_FULL	int	Queue full
POST_QUEUE_WARNING	int	qlen > TRANSMIT_QUEUE_WARNING_LENGTH
POST_SUCCESS	int	Success
Calls		
Function	Where Described	
queue length	Sec. 2.14.4.2.2	
DEBUG CONNECTION	Sec. 2.4.2.4 See Appendix A	
buffer allocate	Sec. 2.14.4.2.12	
buffer enqueue	Sec. 2.14.4.2.18	
rudp_send	Sec. 2.4.2.3.13	
buffer deallocate	Sec. 2.14.4.2.15	

Table 2.4-103: rudp_post_message CSU [8.4.2.3.12]

2.4.2.3.13 rudp_send CSU

This CSU causes an RUDP message to be sent by the underlying UDP layer.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to RUDP_SOCKET	Sec. 2.4.2.4
buffer	pointer to RUDP_HEADER	Sec. 2.4.2.4
buf len	int	Standard
Calls		
Function	Where Described	
udp_write	Sec. 2.4.2.1.5	

Table 2.4-104: rudp_send CSU [8.4.2.3.13]

2.4.2.3.14 has_this_packet_been_acked CSU

This CSU checks whether an acknowledgement for this packet has been received.

Parameters		
Parameters	Type	Where Typedef Declared
rh_ptr	pointer to RUDP_HEADER	Sec. 2.4.2.4
ack	unsigned int	Standard
ReturnValues		
Return Value	Type	Meaning
1	int	Acknowledgement received
0	int	No acknowledgement received

Table 2.4-105: has_this_packet_been_acked CSU [8.4.2.3.14]

2.4.2.3.15 rudp_ack_received CSU

This CSU searches through all of the packets in the retransmission queue and removes all those that have a sequence_id less than or equal to the acknowledgement id received. Acknowledgements are cumulative. An acknowledgement for a packet with a sequence id of "n" is an implicit acknowledgement for all packets with sequence numbers less than "n".

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to RUDP_SOCKET	Sec. 2.4.2.4
ack	unsigned int	Standard
Calls		
Function	Where Described	
DEBUG CONNECTION	Sec. 2.4.2.4 See Appendix A	
buffer traverse and apply	Sec. 2.14.4.2.21	
queue length	Sec. 2.14.4.2.2	
buffer_dequeue	Sec. 2.14.4.2.20	
buffer_deallocate	Sec. 2.14.4.2.15	

Table 2.4-106: rudp_ack_received CSU [8.4.2.3.15]

2.4.2.3.16 rudp_retransmit_buf CSU

This CSU resends a packet.

Parameters		
Parameters	Type	Where Typedef Declared
hp	pointer to RUDP_HEADER	Sec. 2.4.2.4
s	pointer to RUDP_SOCKET	Sec. 2.4.2.4

ReturnValues		
Return Value	Type	Meaning
1	int	Execution was completed
Calls		
Function	Where Described	
DEBUG CONNECTION	Sec. 2.4.2.4 See Appendix A	
rudp_send	Sec. 2.4.2.3.13	
buffer length	Sec. 2.14.4.2.16	

Table 2.4-107: rudp_retransmit_buf CSU [8.4.2.3.16]

2.4.2.3.17 rudp_retransmit CSU

This CSU resends all packets in the retransmission queue (that is, all packets not yet acknowledged). If there are no packets to be retransmitted, a bare ACK (acknowledge) is transmitted so that the Symbolics receives a steady stream of acknowledges.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to RUDP_SOCKET	Sec. 2.4.2.4
Calls		
Function	Where Described	
buffer_traverse_and_apply_n_times	Sec. 2.14.4.2.22	
DEBUG CONNECTION	Sec. 2.4.2.4 See Appendix A	
rudp_send bare_ack	Sec. 2.4.2.3.18	

Table 2.4-108: rudp_retransmit CSU [8.4.2.3.17]

2.4.2.3.18 rudp_send_bare_ack CSU

This CSU sends an RUDP packet with no data in it. It contains an ACK for the last packet received.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to RUDP_SOCKET	Sec. 2.4.2.4

Calls	
Function	Where Described
DEBUG_CONNECTION	Sec. 2.4.2.4 See Appendix A
rudp_send	Sec. 2.4.2.3.13

Table 2.4-109: rudp_send_bare_ack CSU [8.4.2.3.18]

2.4.2.3.19 rudp_tick CSU

This CSU causes retransmissions if the retransmission time has expired.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to RUDP_SOCKET	Sec. 2.4.2.4
Calls		
Function	Where Described	
rudp_ack_received	Sec. 2.4.2.3.15	
rudp_retransmit	Sec. 2.4.2.3.17	

Table 2.4-110: rudp_tick CSU [8.4.2.3.19]

2.4.2.4 rel_conn.h CSU

/simnet/libsrc/libudp/rel_conn.h

This CSU contains the reliable connection layer includes for the Symbolics/MIPS connection. It contains two typedef structures, a number of constant definitions, other external and local definitions, and a macro (DEBUG_CONNECTION(x)) defined in Appendix A.

Item	Type	Where Type Defined
type	int	Standard
seq	int	Standard
ack	int	Standard

Table 2.4-111: RUDP_HEADER Structure Definition

Item	Type	Where Type Defined
connection state	int	Standard
next retransmit time	unsigned int	Standard
next seq out	unsigned int	Standard
last seq in	unsigned int	Standard
last seq ret	unsigned int	Standard
need ack	int	Standard
udp ptr	pointer to UDP CONNECTION	Sec. 2.4.2.2
packets out	int	Standard
packets in	int	Standard
bare acks	int	Standard
packets retransmitted	int	Standard
my packets tossed	int	Standard
sbx packets tossed	int	Standard
marked total bytes out	int	Standard
marked total bytes in	int	Standard
marked new bytes out	int	Standard
marked new bytes in	int	Standard
last marked time	unsigned int	Standard
input buffer	pointer to char	Standard
input ptr	pointer to char	Standard
retransmit queue	pointer to QUEUE	Sec. 2.14.4.3
last synched	int	Standard

Table 2.4-112: RUDP_SOCKET Structure Definition

Constant	Value
RUDP_TYPE SYNCH	1
RUDP_TYPE DATA	2
RUDP_TYPE ACK	3
SHORT RETRANSMIT TIME	500 /* Milliseconds */
NORMAL RETRANSMIT TIME	6000 /* Milliseconds */
CONNECT STATE OPEN	1
CONNECT STATE CLOSED	2
CONNECT STATE LOADED	3
TRANSMIT QUEUE WARNING LENGTH	40
TRANSMIT QUEUE ERROR LENGTH	100
RETRANSMISSION BURST LENGTH	10
POST SUCCESS	0
POST CONNECTION CLOSED	1
POST QUEUE FULL	2
POST QUEUE WARNING	3
D_CONNECTION /* Debug connection bit */	0x00004000

Table 2.4-113: rel_conn.h Constant Definitions

2.4.2.5 buf_conn.c CSC

/simnet/libsrc/libudp/buf_conn.c

The buf_conn.c CSC contains thirteen CSUs (functions) dealing with the Buffered RUDP network layer. These CSUs are shown in Figure 2.4-4. Outgoing packets are buffered together into pieces that just fit into an RUDP packet. This maximizes the amount of useful data in each RUDP packet.

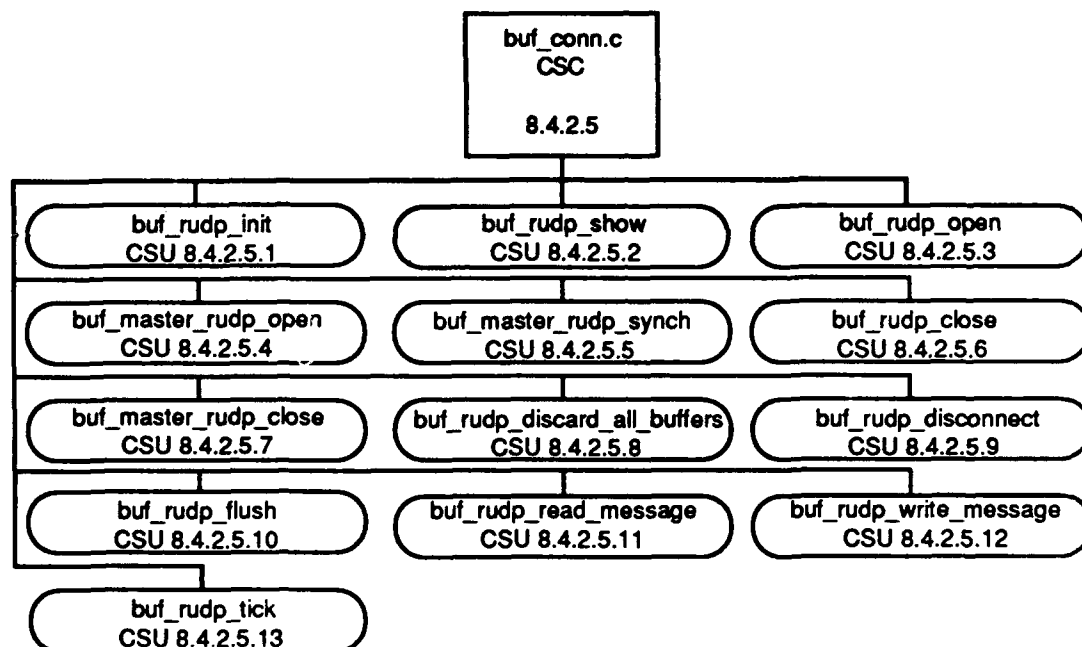


Figure 2.4-4: buf_conn.c CSC Structure

2.4.2.5.1 buf_rudp_init CSU

This CSU initiates the buffered RUDP socket.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to BUF_RUDP_SOCKET	Sec. 2.4.2.6
Calls		
Function	Where Described	
heap allocate	Sec. 2.14.2.1.1	
queue allocate	Sec. 2.14.4.2.1	
rudp init	Sec. 2.4.2.3.1	

Table 2.4-114: buf_rudp_init CSU [8.4.2.5.1]

2.4.2.5.2 buf_rudp_show CSU

This CSU prints statistics on this layer.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to BUF_RUDP_SOCKET	Sec. 2.4.2.6
Calls		
Function	Where Described	
queue_length	Sec. 2.14.4.2.2	
rudp_show	Sec. 2.4.2.3.2	

Table 2.4-115: buff_rudp_show CSU [8.4.2.5.2]

2.4.2.5.3 buf_rudp_open CSU

This CSU opens this layer by initializing the idle timer and opening the RUDP layer.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to BUF_RUDP_SOCKET	Sec. 2.4.2.6
local_addr	pointer to char	Standard
local_port	int	Standard
Calls		
Function	Where Described	
rudp_open	Sec. 2.4.2.3.3	
get_millisecond_time	Sec. 2.14.3.3.2	

Table 2.4-116: buf_rudp_open CSU [8.4.2.5.3]

2.4.2.5.4 buf_master_rudp_open CSU

This CSU is unused.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to BUF_RUDP_SOCKET	Sec. 2.4.2.6
host	pointer to char	Standard
port	int	Standard
ReturnValues		
Return Value	Type	Meaning
-1	int	Connection failed to open

Calls	
Function	Where Described
master_rudp_open	Sec. 2.4.2.3.4
get_millisecond_time	Sec. 2.14.3.3.2

Table 2.4-117: buf_master_rudp_open CSU [8.4.2.5.4]

2.4.2.5.5 buf_master_rudp_synch CSU

This CSU is unused.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to BUF_RUDP_SOCKET	Sec. 2.4.2.6
Calls		
Function	Where Described	
buf_rudp_discard_all_buffers	Sec. 2.4.2.5.8	
master_rudp_synch	Sec. 2.4.2.3.8	

Table 2.4-118: buf_master_rudp_synch CSU [8.4.2.5.5]

2.4.2.5.6 buf_rudp_close CSU

This CSU closes the connection by closing the RUDP layer.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to BUF_RUDP_SOCKET	Sec. 2.4.2.6
Calls		
Function	Where Described	
rudp_close	Sec. 2.4.2.3.5	

Table 2.4-119: buf_rudp_close CSU [8.4.2.5.6]

2.4.2.5.7 buf_master_rudp_close CSU

This CSU is unused.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to BUF_RUDP_SOCKET	Sec. 2.4.2.6

Calls	
Function	Where Described
master_rudp_close	Sec. 2.4.2.3.6

Table 2.4-120: buf_master_rudp_close CSU [8.4.2.5.7]

2.4.2.5.8 buf_rudp_discard_all_buffers CSU

This CSU discards any buffered messages waiting to be retransmitted.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to BUF_RUDP_SOCKET	Sec. 2.4.2.6
Calls		
Function	Where Described	
DEBUG_CONNECTION	Sec. 2.4.2.4 See Appendix A	
buffer_flush	Sec. 2.14.4.2.23	

Table 2.4-121: buf_rudp_discard_all_buffers CSU [8.4.2.5.8]

2.4.2.5.9 buf_rudp_disconnect CSU

This CSU disconnects this connection by disconnecting the RUDP layer. Buffered messages are disabled.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to BUF_RUDP_SOCKET	Sec. 2.4.2.6
Calls		
Function	Where Described	
buf_rudp_discard_all_buffers	Sec. 2.4.2.5.8	
rudp_disconnect	Sec. 2.4.2.3.9	

Table 2.4-122: buf_rudp_disconnect CSU [8.4.2.5.9]

2.4.2.5.10 buf_rudp_flush CSU

This CSU sends the currently buffered messages to the RUDP layer.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to BUF_RUDP_SOCKET	Sec. 2.4.2.6

ReturnValues		
Return Value	Type	Meaning
FALSE	int	Buffer empty
TRUE	int	Buffer was not empty
Calls		
Function	Where Described	
rudp_post_message	Sec. 2.4.2.3.12	

Table 2.4-123: buf_rudp_flush CSU [8.4.2.5.10]

2.4.2.5.11 buff_rudp_read_message CSU

This CSU enters a loop until there is either a valid message for the RUDP layer or there is nothing to read.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to BUF_RUDP_SOCKET	Sec. 2.4.2.6
msg_ptr	pointer to pointer to char	Standard
msg_len	pointer to int	Standard
Calls		
Function	Where Described	
rudp_read_message	Sec. 2.4.2.3.11	
buf_rudp_discard_all_buffers	Sec. 2.4.2.5.8	

Table 2.4-124: buff_rudp_read_message CSU [8.4.2.5.11]

2.4.2.5.12 buf_rudp_write_message CSU

If there is room in the currently buffered message, this CSU adds the input message to the buffered message. Otherwise, the currently buffered message is removed (via buf_rudp_flush) and a new buffered message is started with the input message as the first component.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to BUF_RUDP_SOCKET	Sec. 2.4.2.6
buffer	pointer to char	Standard
buf_len	int	Standard
Calls		
Function	Where Described	
buf_rudp_flush	Sec. 2.4.2.5.10	

Table 2.4-125: buf_rudp_write_message CSU [8.4.2.5.12]

2.4.2.5.13 buf_rudp_tick CSU

This CSU pulls messages off the input queue and writes these messages via `buf_rudp_write_message`. If the idle time has elapsed, the currently buffered message is sent, preventing a buffered message from waiting indefinitely for enough messages to fill the RUDP packet before it is sent. The RUDP layer is ticked.

Parameters		
Parameters	Type	Where Typedef Declared
s	pointer to BUF_RUDP_SOCKET	Sec. 2.4.2.6
Calls		
Function	Where Described	
queue length	Sec. 2.14.4.1.3	
buffer dequeue	Sec. 2.14.4.2.20	
buf_rudp_write_message	Sec. 2.4.2.5.12	
buffer length	Sec. 2.14.4.2.16	
buffer deallocate	Sec. 2.14.4.2.15	
DEBUG_CONNECTION	Sec. 2.4.2.4 See Appendix A	
get_millisecond_time	Sec. 2.14.3.3.2	
buf_rudp_flush	Sec. 2.4.2.5.10	
rudp_tick	Sec. 2.4.2.3.19	

Table 2.4-126: `buf_rudp_tick` CSU [8.4.2.5.13]**2.4.2.6 buf_conn.h CSU**

/simnet/libsrc/libudp/buf_conn.h

This CSU consists of the buffered connection includes. These includes consist of a typedef struct and two constant defines.

The typedef struct is tagged `buf_rudp_socket`.

Item	Type	Where Type Defined
send_buffer_offset	int	Standard
next_idle_time	unsigned int	Standard
message_in	int	Standard
message_out	int	Standard
idle_flushes	int	Standard
send_buffer	pointer to char	Standard
input_queue	pointer to QUEUE	Sec. 2.14.4.3
rudp_skt	pointer to RUDP_SOCKET	Sec. 2.4.2.4

Table 2.4-127: `BUF_RUDP_SOCKET` Structure Definition

Constant	Value
BUF_RUDP_IDLE_DURATION	1000
INPUT_QUEUE_WARNING_LENGTH	256

Table 2.4-128: `buf_conn.h` Constant Definitions

2.4.3 SAF Command Processor CSC

2.4.3.1 map.c CSC

/simnet/src/host/map.c

This file handles the symbol mapping between the workstation and Simhost. The map.c CSC structure, consisting of nine CSUs, is depicted in Figure 2.4-5.

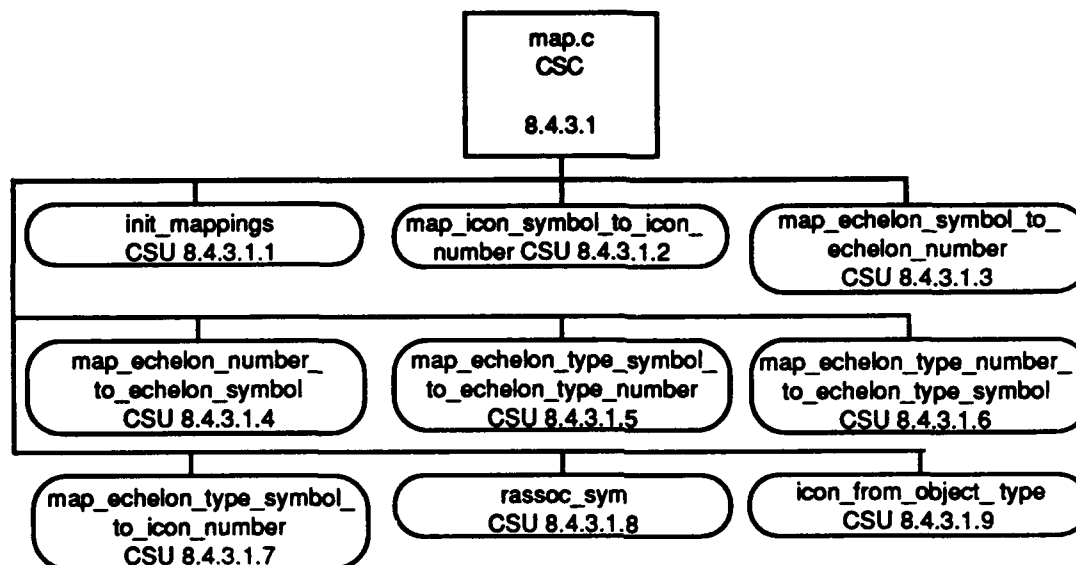


Figure 2.4-5: map.c CSC Structure

2.4.3.1.1 init_mappings CSU

This CSU reads the mapping file and initializes the various mappings.

Parameters		
Parameters	Type	Where Typedef Declared
filename	pointer to char	Standard
Errors		
Error Name	Reason for Error	
Mapping file not found.	!reader_read_file(filename.&db)	
Calls		
Function	Where Described	
reader_read_file	Sec. 2.1.1.1.7	
ERROR_OUT	Sec. 2.5.2.2	
abort		
find_tag	Sec. 2.1.1.4.3	
map_echelon_symbol_to_echelon_number	Sec. 2.4.3.1.3	

Table 2.4-129: init_mappings CSU [8.4.3.1.1]

2.4.3.1.2 map_icon_symbol_to_icon_number CSU

This CSU maps an icon name into an icon number.

Parameters		
Parameters	Type	Where Typedef Declared
sym	pointer to char	Standard
ReturnValues		
Return Value	Type	Meaning
0	unsigned short	Icon name unknown
result[2].integer	unsigned short	Icon number
Calls		
Function	Where Described	
find_tag	Sec. 2.1.1.4.3	

Table 2.4-130: map_icon_symbol_to_icon_number CSU [8.4.3.1.2]

2.4.3.1.3 map_echelon_symbol_to_echelon_number CSU

This CSU maps an echelon name into an echelon number.

Parameters		
Parameters	Type	Where Typedef Declared
sym	pointer to char	Standard
ReturnValues		
Return Value	Type	Meaning
0	unsigned short	Echelon name unknown
result[2].integer	unsigned short	Echelon number
Calls		
Function	Where Described	
find_tag	Sec. 2.1.1.4.3	

Table 2.4-131: map_echelon_symbol_to_echelon_number CSU [8.4.3.1.3]

2.4.3.1.4 map_echelon_number_to_echelon_symbol CSU

This CSU maps an echelon number into an echelon name.

Parameters		
Parameters	Type	Where Typedef Declared
num	unsigned char	Standard

ReturnValues		
Return Value	Type	Meaning
rassoc_sym(num,echelon_map)	pointer to char	Pointer to echelon name
Calls		
Function	Where Described	
rassoc_sym	Sec. 2.4.3.1.8	

Table 2.4-132: map_echelon_number_to_echelon_symbol CSU [8.4.3.1.4]

2.4.3.1.5 map_echelon_type_symbol_to_echelon_type_number CSU

This CSU maps an echelon type name into an echelon type number

Parameters		
Parameters	Type	Where Typedef Declared
sym	pointer to char	Standard
ReturnValues		
Return Value	Type	Meaning
0	unsigned short	Echelon type name unknown
result[2].integer	unsigned short	Echelon type number
Calls		
Function	Where Described	
find_tag	Sec. 2.1.1.4.3	

Table 2.4-133: map_echelon_type_symbol_to_echelon_type_number CSU [8.4.3.1.5]

2.4.3.1.6 map_echelon_type_number_to_echelon_type_symbol CSU

This CSU maps an echelon type number into an echelon type name.

Parameters		
Parameters	Type	Where Typedef Declared
num	unsigned char	Standard
ReturnValues		
Return Value	Type	Meaning
rassoc_sym(num,echelon_type_map)	pointer to char	Pointer to echelon type name

Calls	
Function	Where Described
rassoc_sym	Sec. 2.4.3.1.8

Table 2.4-134: map_echelon_type_number_to_echelon_type_symbol CSU [8.4.3.1.6]

2.4.3.1.7 map_echelon_type_symbol_to_icon_number CSU

This CSU maps an echelon type name into an icon number.

Parameters		
Parameters	Type	Where Typedef Declared
sym	pointer to char	Standard
ReturnValues		
Return Value	Type	Meaning
0	unsigned short	Echelon type name unknown
map_icon_symbol_to_icon_number(result[2].charptr)	unsigned short	Icon number
Calls		
Function	Where Described	
find_tag	Sec. 2.1.1.4.3	
map_icon_symbol_to_icon_number	Sec. 2.4.3.1.2	

Table 2.4-135: map_echelon_type_symbol_to_icon_number CSU [8.4.3.1.7]

2.4.3.1.8 rassoc_sym CSU

This CSU does an inverse association from the mapping file. It finds the name that maps to the input number.

Parameters		
Parameters	Type	Where Typedef Declared
num	unsigned char	Standard
table	pointer to DATA_UNION	
ReturnValues		
Return Value	Type	Meaning
entry[1].charptr	pointer to char	Pointer to the name
UNKNOWN_SYM	pointer to char	No name found

Table 2.4-136: rassoc_sym CSU [8.4.3.1.8]

2.4.3.1.9 icon_from_object_type CSU

This CSU classifies objects of the input object type into a valid icon number.

Parameters		
Parameters	Type	Where Typedef Declared
obj	ObjectType	p_sim.h
ReturnValues		
Return Value	Type	Meaning
0	unsigned short	Unknown input
map_icon_symbol_to_icon_number(FWA_SYM)	unsigned short	Icon number of Fixed Wing symbol
map_icon_symbol_to_icon_number(SCOUT_RWA_SYM)	unsigned short	Icon number of Scout Rotary Wing symbol
map_icon_symbol_to_icon_number(ATTACK_RWA_SYM)	unsigned short	Icon number of Attack Rotary Wing symbol
map_icon_symbol_to_icon_number(ADA_SYM)	unsigned short	Icon number of Anti Aircraft symbol
map_icon_symbol_to_icon_number(MECH_SYM)	unsigned short	Icon number of Personnel Carrier or Ground Recon symbol
map_icon_symbol_to_icon_number(CP_SYM)	unsigned short	Icon number of Command Post symbol
map_icon_symbol_to_icon_number(SPH_SYM)	unsigned short	Icon number of Howitzer symbol
map_icon_symbol_to_icon_number(MORTAR_SYM)	unsigned short	Icon number of Mortar symbol
map_icon_symbol_to_icon_number(SUPPLY_SYM)	unsigned short	Icon number of Recovery symbol
map_icon_symbol_to_icon_number(AMMO_SYM)	unsigned short	Icon number of ammo Supply Truck symbol
map_icon_symbol_to_icon_number(FUEL_SYM)	unsigned short	Icon number of fuel Supply Truck symbol
map_icon_symbol_to_icon_number(TANK_SYM)	unsigned short	Icon number of Main Battle Tank symbol
Calls		
Function	Where Described	
map_icon_symbol_to_icon_number	Sec. 2.4.3.1.2	

Table 2.4-137: icon_from_object_type CSU [8.4.3.1.9]

2.4.3.2 sbx.c CSC

/simnet/src/host/sbx.c

This CSC contains all the code that handles communications with the workstation.

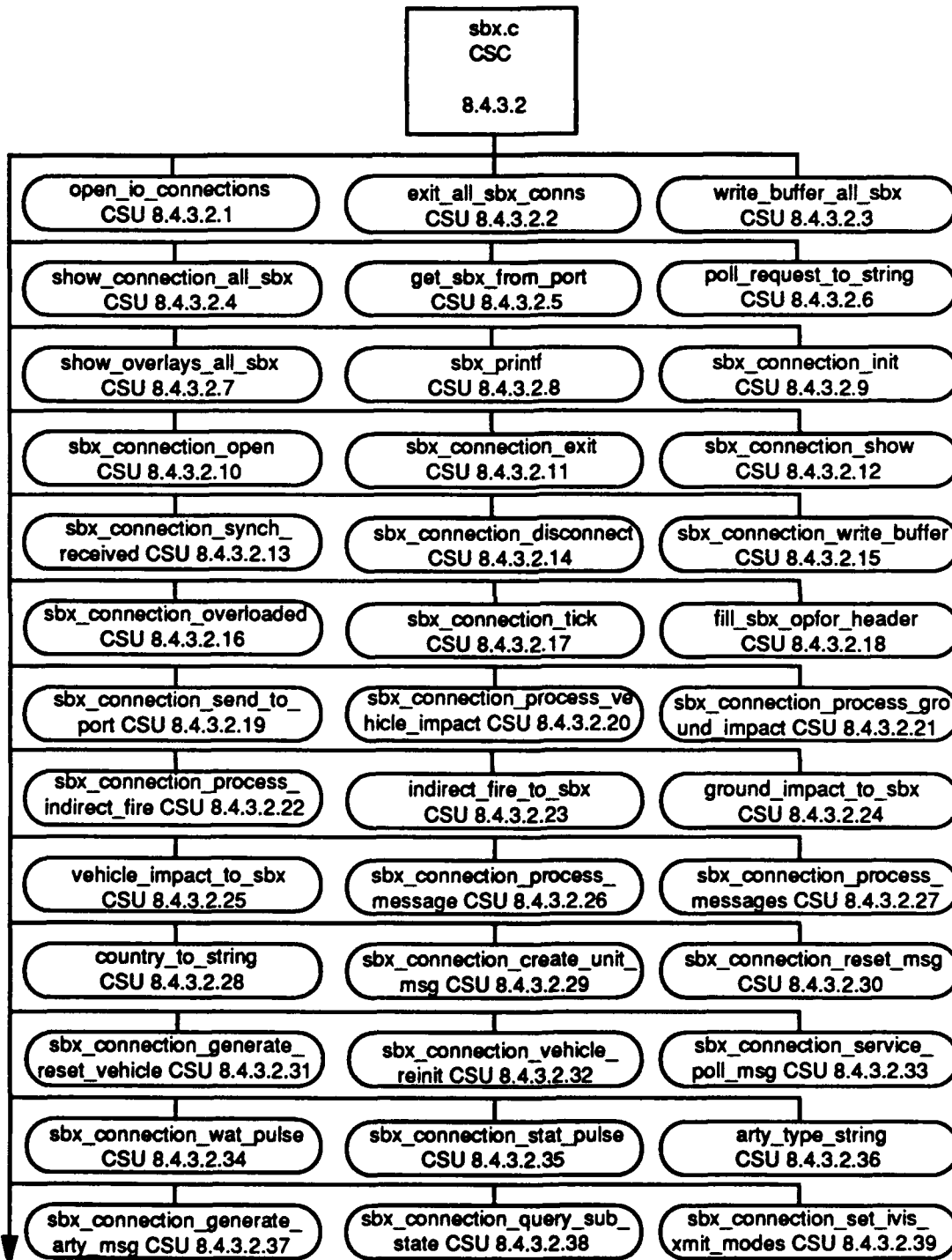


Figure 2.4-6: sbx.c CSC Structure Part 1 of 2



Figure 2.4-7: sbx.c CSC Structure Part 2 of 2

In addition to the 79 CSUs, this CSC contains a macro, ORDER_FROM_USER defined in Appendix A.

2.4.3.2.1 open_io_connections CSU

This CSU opens all the specified ports and schedules the sbx_connection tick to be ticked every 100 milliseconds.

Parameters		
Parameters	Type	Where Typedef Declared
port_array[]	int	Standard
Calls		
Function	Where Described	
allocate sbx connection	Sec. 2.4.3.3	
sbx connection init	Sec. 2.4.3.2.9	
sbx connection open	Sec. 2.4.3.2.10	
get local address	Sec. 2.4.2.1.6	
periodic incl	Sec. 2.2.1.1.2	

Table 2.4-138: open_io_connections CSU [8.4.3.2.1]

2.4.3.2.2 exit_all_sbx_conns CSU

This CSU causes all ports to be closed.

Calls	
Function	Where Described
sbx connection exit	Sec. 2.4.3.2.11

Table 2.4-139: exit_all_sbx_conns CSU [8.4.3.2.2]

2.4.3.2.3 write_buffer_all_sbx CSU

This CSU writes a message to all open ports.

Parameters		
Parameters	Type	Where Typedef Declared
bufp	pointer to OPFOR_HEADER	Sec. 2.4.1.1
Calls		
Function	Where Described	
sbx connection write buffer	Sec. 2.4.3.2.15	

Table 2.4-140: write_buffer_all_sbx CSU [8.4.3.2.3]

2.4.3.2.4 show_connection_all_sbx CSU

This CSU causes all open ports to display status information.

Calls	
Function	Where Described
sbx_connection_show	Sec. 2.4.3.2.12

Table 2.4-141: show_connection_all_sbx CSU [8.4.3.2.4]

2.4.3.2.5 get_sbx_from_port CSU

This CSU returns the SBX_CONNECTION_VARS corresponding to the given port id.

Parameters		
Parameters	Type	Where Typedef Declared
port	int	Standard
ReturnValues		
Return Value	Type	Meaning
sbx_connections[i]	pointer to SBX_CONNECTION_VARS	Pointer to block of connection variables
NULL	pointer to SBX_CONNECTION_VARS	Port id unknown

Table 2.4-142: get_sbx_from_port CSU [8.4.3.2.5]

2.4.3.2.6 poll_request_to_string CSU

This CSU returns a human readable string identifying what type of poll request has been received.

Parameters		
Parameters	Type	Where Typedef Declared
req	int	Standard
ReturnValues		
Return Value	Type	Meaning
"Gods Eye View"	pointer to char	God's view poll request received
"Commanders Eye View"	pointer to char	Commander's view poll request received
"Specific Request"	pointer to char	Specific poll request received
"Unknown request"	pointer to char	Unknown poll request received

Table 2.4-143: poll_request_to_string CSU [8.4.3.2.6]

2.4.3.2.7 show_overlays_all_sbx CSU

This CSU causes all overlays associated with each port to be displayed.

Calls	
Function	Where Described
show_overlays	Sec. 2.10.2.1.34

Table 2.4-144: show_overlays_all_sbx CSU [8.4.3.2.7]

2.4.3.2.8 sbx_printf CSU

This CSU causes a string message, in the form of a printf string with arguments, to be sent to the specified port.

Parameters		
Parameters	Type	Where Typedef Declared
vid	unsigned int	Standard
message_type	unsigned short	Standard
port	int	Standard
format_string	pointer to char	Standard
Calls		
Function	Where Described	
buffer allocate	Sec. 2.14.4.2.12	
fill_sbx_opfor_header	Sec. 2.4.3.2.18	
sbx_connection_send_to_port	Sec. 2.4.3.2.19	
buffer deallocate	Sec. 2.14.4.2.15	

Table 2.4-145: sbx_printf CSU [8.4.3.2.8]

2.4.3.2.9 sbx_connection_init CSU

This CSU initializes all the data structures for a port. The view is initialized to be commander.

Parameters		
Parameters	Type	Where Typedef Declared
sbx_connection	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
Calls		
Function	Where Described	
sbx_connection_stat_pulse	Sec. 2.4.3.2.35	
heap allocate	Sec. 2.14.2.1.1	
allocate_bitfield	Sec. 2.14.3.4.1	
clear_bitfield	Sec. 2.14.3.4.6	
buf_rudp_init	Sec. 2.4.2.5.1	

Table 2.4-146: sbx_connection_init CSU [8.4.3.2.9]

2.4.3.2.10 sbx_connection_open CSU

This CSU opens the port associated with an sbx_connection.

Parameters		
Parameters	Type	Where Typedef Declared
sbx_connection	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
local_addr	pointer to char	Standard
local_port	int	Standard
Calls		
Function	Where Described	
buf_rudp_open	Sec. 2.4.2.5.3	

Table 2.4-147: sbx_connection_open CSU [8.4.3.2.10]

2.4.3.2.11 sbx_connection_exit CSU

The CSU closes the port associated with an sbx_connection.

Parameters		
Parameters	Type	Where Typedef Declared
sbx_connection	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
Calls		
Function	Where Described	
buf_rudp_close	Sec. 2.4.2.5.6	
DEBUG_CONNECTION	Sec. 2.4.2.4 See Appendix A	

Table 2.4-148: sbx_connection_exit CSU [8.4.3.2.11]

2.4.3.2.12 sbx_connection_show CSU

This CSU causes various statistics concerning an sbx_connection to be printed.

Parameters		
Parameters	Type	Where Typedef Declared
sbx_connection	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
Errors		
Error Name	Reason for Error	
connection in unknown state	SBX connection is neither open, nor closed, nor loaded.	

Calls	
Function	Where Described
poll request to string	Sec. 2.4.3.2.6
forceID to string	Sec. 2.9.2.1.3
sbx_connection_show_top_level units	Sec. 2.4.3.2.79
buf_rudp_show	Sec. 2.4.2.5.2
ERROR_OUT	Sec. 2.5.2.2

Table 2.4-149: sbx_connection_show CSU [8.4.3.2.12]

2.4.3.2.13 sbx_connection_synch_received CSU

This CSU causes a MACHINE_STATUS message to be sent to the Symbolics when a connection is established. A deferred function is started to cause the sending of PAE messages.

Parameters		
Parameters	Type	Where Typedef Declared
sbx_connection	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
Calls		
Function	Where Described	
sbx_connection_stat_pulse	Sec. 2.4.3.2.35	
buffer_allocate	Sec. 2.14.4.2.12	
fill_sbx_opfor_header	Sec. 2.4.3.2.18	
get_millisecond_time	Sec. 2.14.3.3.2	
sbx_connection_write_buffer	Sec. 2.4.3.2.15	
buffer_deallocate	Sec. 2.14.4.2.15	
vehicle_iterator_reset	Sec. 2.9.3.1.1	
deferred_incl	Sec. 2.2.1.1.1	

Table 2.4-150: sbx_connection_synch_received CSU [8.4.3.2.13]

2.4.3.2.14 sbx_connection_disconnect CSU

This CSU cancels any deferred functions associated with connection and disconnects the buf_RUDP layer.

Parameters		
Parameters	Type	Where Typedef Declared
sbx_connection	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3

Calls	
Function	Where Described
cancel fncl	Sec. 2.2.1.1.4
buf_rudp_disconnect	Sec. 2.4.2.5.9

Table 2.4-151: sbx_connection_disconnect CSU [8.4.3.2.14]

2.4.3.2.15 sbx_connection_write_buffer CSU

This CSU enqueues a message on the input queue of the associated buf_RUDP layer.

Parameters		
Parameters	Type	Where Typedef Declared
sbx_connection	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
bufp	pointer to OPFOR_HEADER	Sec. 2.4.1.1
Errors		
Error Name	Reason for Error	
connection in unknown state	SBX connection is neither open, nor closed, nor loaded.	
Calls		
Function	Where Described	
buffer enqueue	Sec. 2.14.4.2.18	
ERROR OUT	Sec. 2.5.2.2	

Table 2.4-152: sbx_connection_write_buffer CSU [8.4.3.2.15]

2.4.3.2.16 sbx_connection_overloaded CSU

This CSU returns information about whether or not a connection is overloaded. A connection is overloaded if there is a large backup of messages to be sent or a large backup of sent messages waiting for acknowledgements.

Parameters		
Parameters	Type	Where Typedef Declared
sbx_connection	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
ReturnValues		
Return Value	Type	Meaning
TRUE	int	Input queue or retransmit queue is filled to the warning point.
FALSE	int	No overload.

Calls	
Function	Where Described
queue_length	Sec. 2.14.4.2.2

Table 2.4-153: sbx_connection_overloaded CSU [8.4.3.2.16]

2.4.3.2.17 sbx_connection_tick CSU

This CSU reads messages from the buf_RUDP layer and causes the messages to be processed.

Parameters		
Parameters	Type	Where Typedef Declared
sbx_connection	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
Calls		
Function	Where Described	
buf_rudp_read_message	Sec. 2.4.2.5.11	
sbx_connection_synch_received	Sec. 2.4.3.2.13	
sbx_connection_process_messages	Sec. 2.4.3.2.27	
buf_rudp_tick	Sec. 2.4.2.5.13	

Table 2.4-154: sbx_connection_tick CSU [8.4.3.2.17]

2.4.3.2.18 fill_sbx_opfor_header CSU

This CSU fills out the header information for an outgoing packet.

Parameters		
Parameters	Type	Where Typedef Declared
msg_ptr	pointer to OPFOR_HEADER	Sec. 2.4.1.1
id	unsigned int	Standard
type	int	Standard

Table 2.4-155: fill_sbx_opfor_header CSU [8.4.3.2.18]

2.4.3.2.19 sbx_connection_send_to_port CSU

This CSU sends a message to the connection associated with the specified port.

Parameters		
Parameters	Type	Where Typedef Declared
port_number	int	Standard
bufp	pointer to OPFOR_HEADER	Sec. 2.4.1.1

Calls	
Function	Where Described
sbx_connection write buffer	Sec. 2.4.3.2.15
get_sbx_from_port	Sec. 2.4.3.2.5

Table 2.4-156: sbx_connection_send_to_port CSU [8.4.3.2.19]

2.4.3.2.20 sbx_connection_process_vehicle_impact CSU

This CSU sends a vehicle impact message to a connection if the connection has been previously informed of the attacker and target vehicles.

Parameters		
Parameters	Type	Where Typedef Declared
sbx_connection	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
spdu	pointer to SimulationPDU	p_sim.h
Calls		
Function	Where Described	
saf_id_from_simnet_id		
extract_bit		
buffer_allocate	Sec. 2.14.4.2.12	
fill_sbx_opfor_header	Sec. 2.4.3.2.18	
sbx_connection_write_buffer	Sec. 2.4.3.2.15	
buffer_deallocate	Sec. 2.14.4.2.15	

Table 2.4-157: sbx_connection_process_vehicle_impact CSU [8.4.3.2.20]

2.4.3.2.21 sbx_connection_process_ground_impact CSU

This CSU sends a ground impact message to a connection.

Parameters		
Parameters	Type	Where Typedef Declared
sbx_connection	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
spdu	pointer to SimulationPDU	p_sim.h

Calls	
Function	Where Described
saf id from simnet id	
extract bit	
buffer allocate	Sec. 2.14.4.2.12
fill sbx opfor header	Sec. 2.4.3.2.18
sbx connection write buffer	Sec. 2.4.3.2.15
buffer deallocate	Sec. 2.14.4.2.15

Table 2.4-158: sbx_connection_process_ground_impact CSU [8.4.3.2.21]

2.4.3.2.22 sbx_connection_process_indirect_fire CSU

This CSU sends an indirect fire message to a connection.

Parameters		
Parameters	Type	Where Typedef Declared
sbx_connection	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
spdu	pointer to SimulationPDU	p_sim.h
Calls		
Function	Where Described	
buffer allocate	Sec. 2.14.4.2.12	
fill sbx opfor header	Sec. 2.4.3.2.18	
sbx connection write buffer	Sec. 2.4.3.2.15	
buffer deallocate	Sec. 2.14.4.2.15	

Table 2.4-159: sbx_connection_process_indirect_fire CSU [8.4.3.2.22]

2.4.3.2.23 indirect_fire_to_sbx CSU

This CSU calls sbx_connection_process_indirect_fire for each connection.

Parameters		
Parameters	Type	Where Typedef Declared
spdu	pointer to SimulationPDU	p_sim.h
Calls		
Function	Where Described	
sbx_connection_process_indirect_fire	Sec. 2.4.3.2.22	

Table 2.4-160: indirect_fire_to_sbx CSU [8.4.3.2.23]

2.4.3.2.24 ground_impact_to_sbx CSU

This CSU calls `sbx_connection_process_ground_impact` for each connection.

Parameters		
Parameters	Type	Where Typedef Declared
<code>spdu</code>	pointer to SimulationPDU	<code>p_sim.h</code>
Calls		
Function	Where Described	
<code>sbx_connection_process_ground_impact</code>	Sec. 2.4.3.2.21	

Table 2.4-161: ground_impact_to_sbx CSU [8.4.3.2.24]

2.4.3.2.25 vehicle_impact_to_sbx CSU

This CSU calls `sbx_connection_process_vehicle_impact` for each connection.

Parameters		
Parameters	Type	Where Typedef Declared
<code>spdu</code>	pointer to SimulationPDU	<code>p_sim.h</code>
Calls		
Function	Where Described	
<code>sbx_connection_process_vehicle_impact</code>	Sec. 2.4.3.2.20	

Table 2.4-162: vehicle_impact_to_sbx CSU [8.4.3.2.25]

2.4.3.2.26 sbx_connection_process_message CSU

This CSU, dependent on the message type, calls the appropriate handler to process that particular message type.

Parameters		
Parameters	Type	Where Typedef Declared
<code>sbx_connection</code>	pointer to SBX CONNECTION VARS	Sec. 2.4.3.3
<code>msg_ptr</code>	pointer to OPFOR_GENERIC_MSG	Sec. 2.4.1.1
ReturnValues		
Return Value	Type	Meaning
<code>sizeof(...)</code>	int	Size of the message in bytes.

Errors	
Error Name	Reason for Error
Received obsolete CONTINUE_MISSION message	msg_type was set to the obsolete value OPFOR_MSG_CONTINUE_MISSION
Unknown message type	msg_type was set to an unknown value
Calls	
Function	Where Described
print opfor header	Sec. 2.4.3.2.76
sbx_connection_create_unit_ msg	Sec. 2.4.3.2.29
sbx_connection_reset msg	Sec. 2.4.3.2.30
sbx_connection_generate_ arty_msg	Sec. 2.4.3.2.37
sbx_connection_vehicle_ reinit	Sec. 2.4.3.2.32
sbx_connection_service_ poll msg	Sec. 2.4.3.2.33
sbx_connection_disconnect	Sec. 2.4.3.2.14
sbx_connection_query_sub_ state	Sec. 2.4.3.2.38
sbx_connection_set_ivis_ xmit_modes	Sec. 2.4.3.2.39
sbx_connection_set_ivis_ parameters	Sec. 2.4.3.2.40
ERROR OUT	Sec. 2.5.2.2
sbx_add point	Sec. 2.4.3.2.45
sbx_add area	Sec. 2.4.3.2.46
sbx_add zone	Sec. 2.4.3.2.47
sbx_add line	Sec. 2.4.3.2.48
sbx_add route	Sec. 2.4.3.2.49
sbx_delete overlay	Sec. 2.4.3.2.50
sbx_execute overlay	Sec. 2.4.3.2.51
sbx_halt	Sec. 2.4.3.2.52
sbx_change speed	Sec. 2.4.3.2.53
sbx_change formation	Sec. 2.4.3.2.54
sbx_delete cm	Sec. 2.4.3.2.55
sbx_follow vehicle	Sec. 2.4.3.2.57
sbx_goto point	Sec. 2.4.3.2.59
sbx_resume mission	Sec. 2.4.3.2.60
sbx_face direction	Sec. 2.4.3.2.61
sbx_set_targeting_ parameters	Sec. 2.4.3.2.62
sbx_attach stealth	Sec. 2.4.3.2.70
sbx_rejoin unit	Sec. 2.4.3.2.56
sbx_simulator in command	Sec. 2.4.3.2.58
sbx_resupply	Sec. 2.4.3.2.71

Table 2.4-163 continued on the following page

Function	Where Described
sbx_land	Sec. 2.4.3.2.63
sbx_altitude	Sec. 2.4.3.2.65
sbx_hold	Sec. 2.4.3.2.67
sbx_attack	Sec. 2.4.3.2.69

Table 2.4-163: sbx_connection_process_message CSU [8.4.3.2.26]

2.4.3.2.27 sbx_connection_process_messages CSU

This CSU processes all messages through multiple calls to sbx_connection_process_message.

Parameters		
Parameters	Type	Where Typedef Declared
sbx_connection	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
msg_ptr	pointer to char	Standard
msg_len	int	Standard
Calls		
Function	Where Described	
sbx_connection_process_message	Sec. 2.4.3.2.26	

Table 2.4-164: sbx_connection_process_messages CSU [8.4.3.2.27]

2.4.3.2.28 country_to_string CSU

This CSU returns a human readable string identifying a country code.

Parameters		
Parameters	Type	Where Typedef Declared
c	unsigned char	Standard
ReturnValues		
Return Value	Type	Meaning
"Other"	char	Country is not USA or USSR
"USA"	char	Country is USA
"USSR"	char	Country is USSR
"unknown"	char	Country is unknown

Table 2.4-165: country_to_string CSU [8.4.3.2.28]

2.4.3.2.29 sbx_connection_create_unit_msg CSU

This CSU processes a create unit message and causes the specified unit to be created.

Parameters		
Parameters	Type	Where Typedef Declared
sbx_connection	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
msg_ptr	pointer to CREATE_MSG	Sec. 2.4.1.1
Calls		
Function	Where Described	
map_echelon_number_to_echelon_symbol	Sec. 2.4.3.1.4	
map_echelon_type_number_to_echelon_type_symbol	Sec. 2.4.3.1.6	
forceID to string	Sec. 2.9.2.1.3	
country to string	Sec. 2.4.3.2.28	
print message position	Sec. 2.4.3.2.77	
create unit	Sec. 2.11.1.2	
get_symbol	Sec. 2.1.1.3.2	

Table 2.4-166: sbx_connection_create_unit_msg CSU [8.4.3.2.29]

2.4.3.2.30 sbx_connection_reset_msg CSU

This CSU processes a reset message and removes the specified vehicles. Outstanding deferred functions are canceled, overlays associated with the connection are removed and a deferred function, to inform the connection what units are left, is set up.

Parameters		
Parameters	Type	Where Typedef Declared
sbx_connection	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
msg_ptr	pointer to RESET_MSG	Sec. 2.4.1.1
Errors		
Error Name	Reason for Error	
SBX_CONN in unknown state	SBX connection is neither open, nor closed, nor loaded.	

Table 2.4-167: sbx_connection_reset_msg CSU [8.4.3.2.30]

2.4.3.2.31 sbx_connection_generate_reset_vehicle CSU

This CSU generates a reset message for a vehicle and processes this message. This function is called when remote vehicles are deactivated or timed-out from the battlefield.

Parameters		
Parameters	Type	Where Typedef Declared
sbx_connection	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
vehicle	unsigned int	Standard
Calls		
Function	Where Described	
fill_sbx_opfor_header	Sec. 2.4.3.2.18	
sbx_connection_reset_msg	Sec. 2.4.3.2.30	

Table 2.4-168: sbx_connection_generate_reset_vehicle CSU [8.4.3.2.31]

2.4.3.2.32 sbx_connection_vehicle_reinit CSU

This CSU processes a vehicle reinitialize message and causes a vehicle's position, bearing, fuel and ammunition to be initialized to specified values.

Parameters		
Parameters	Type	Where Typedef Declared
sbx_connection	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
msg_ptr	pointer to VEHICLE_REINIT_MSG	Sec. 2.4.1.1
Calls		
Function	Where Described	
LOOKUP_VEHICLE	Sec. 2.9.3.2	
saf_vehicle_reinit	Sec. 2.6.1.1.17	

Table 2.4-169: sbx_connection_vehicle_reinit CSU [8.4.3.2.32]

2.4.3.2.33 sbx_connection_service_poll_msg CSU

This CSU processes a POLL_MSG. Three types of poll views are supported. In GODS_EYE_VIEW, all vehicles are reported. In COMMANDERS_EYE_VIEW, all vehicles known to vehicles created on the workstation are reported. In NON_GODS_EYE_VIEW, all vehicles known by a unit are reported. A deferred function is set to begin sending the position messages for the desired vehicles.

Parameters		
Parameters	Type	Where Typedef Declared
sbx_connection	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
msg_ptr	pointer to POLL_MSG	Sec. 2.4.1.1

Errors	
Error Name	Reason for Error
Unknown poll request	Neither GODS_EYE_VIEW, nor COMMANDERS_EYE_VIEW, nor NON_GODS_EYE_VIEW was requested.
Calls	
Function	Where Described
sbx_connection_wat_pulse	Sec. 2.4.3.2.34
DEBUG CONNECTION	Sec. 2.4.2.4 See Appendix A
poll request to string	Sec. 2.4.3.2.6
sbx_set_all_known_vehicles	Sec. 2.4.3.2.42
sbx_set_top_level_known_vehicles	Sec. 2.4.3.2.44
sbx_set_specific_known_vehicles	Sec. 2.4.3.2.43
ERROR OUT	Sec. 2.5.2.2
deferred_fnc1	Sec. 2.2.1.1.1

Table 2.4-170: sbx_connection_service_poll_msg CSU [8.4.3.2.33]

2.4.3.2.34 sbx_connection_wat_pulse CSU

This CSU sends position messages for the desired vehicles.

Parameters		
Parameters	Type	Where Typedef Declared
sbx_connection	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
wat_ptr	pointer to VEHICLE_POSITION_MSG	Sec. 2.4.1.1
Calls		
Function	Where Described	
sbx_connection_overloaded	Sec. 2.4.3.2.16	
deferred_fnc1	Sec. 2.2.1.1.1	
sbx_connection_write_buffer	Sec. 2.4.3.2.15	
buffer_deallocate	Sec. 2.14.4.2.15	
LOOKUP_VEHICLE	Sec. 2.9.3.2	
type_ok		
extract_bit		
buffer_allocate	Sec. 2.14.4.2.12	
fill_sbx_opfor_header	Sec. 2.4.3.2.18	
fill_in_position_data	Sec. 2.14.1.1.3	
saf_vehicle_checkpoint_state	Sec. 2.6.1.1.16	
buf_rudp_tick	Sec. 2.4.2.5.13	
buf_rudp_flush	Sec. 2.4.2.5.10	

Table 2.4-171: sbx_connection_wat_pulse CSU [8.4.3.2.34]

2.4.3.2.35 sbx_connection_stat_pulse CSU

This CSU sends PAE messages for all the vehicles.

Parameters		
Parameters	Type	Where Typedef Declared
sbx_connection	pointer to SBX CONNECTION VARS	Sec. 2.4.3.3
vs_ptr	pointer to VEHICLE PAE MSG	Sec. 2.4.1.1
Calls		
Function	Where Described	
sbx_connection overloaded	Sec. 2.4.3.2.16	
deferred_fnc1	Sec. 2.2.1.1.1	
sbx_connection write buffer	Sec. 2.4.3.2.15	
buffer deallocate	Sec. 2.14.4.2.15	
vehicle iterator once next	Sec. 2.9.3.1.3	
buffer allocate	Sec. 2.14.4.2.12	
fill sbx_opfor header	Sec. 2.4.3.2.18	
OBJ VEHICLEID	Sec. 2.9.1.1	
fill in position data	Sec. 2.14.1.1.3	
fill in appearance data	Sec. 2.14.1.1.1	
fill in echelon data	Sec. 2.14.1.1.2	

Table 2.4-172: sbx_connection_stat_pulse CSU [8.4.3.2.35]

2.4.3.2.36 arty_type_string CSU

This CSU returns a human-readable string corresponding to an artillery type.

Parameters		
Parameters	Type	Where Typedef Declared
type	int	Standard
ReturnValues		
Return Value	Type	Meaning
"ARTY TYPE GROUND"	char	Ground artillery
"ARTY TYPE VEHICLE"	char	Vehicle artillery
"ARTY TYPE DEATH"	char	Death artillery
"UNKNOWN"	char	Unknown artillery

Table 2.4-173: arty_type_string CSU [8.4.3.2.36]

2.4.3.2.37 sbx_connection_generate_arty_msg CSU

This CSU processes an artillery message and causes artillery blasts to be generated.

Parameters		
Parameters	Type	Where Typedef Declared
msg_ptr	pointer to ARTY_MSG	Sec. 2.4.1.1
Errors		
Error Name	Reason for Error	
Can't kill nonexistent vehicle	Victim does not exist	
Calls		
Function	Where Described	
arty_type_string	Sec. 2.4.3.2.36	
LOOKUP_VEHICLE	Sec. 2.9.3.2	
ERROR_OUT	Sec. 2.5.2.2	
vehicle_kill	Sec. 2.3.1.2	
generate indirect fire_pkt	Sec. 2.3.1.4	

Table 2.4-174: sbx_connection_generate_arty_msg CSU [8.4.3.2.37]

2.4.3.2.38 sbx_connection_query_sub_state CSU

This CSU processes a query substate message requesting more information about a vehicle or composite, and causes a report to be sent out by a unit.

Parameters		
Parameters	Type	Where Typedef Declared
sbx	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
msg_ptr	pointer to QUERY_SUB_STATE_MSG	Sec. 2.4.1.1
Errors		
Error Name	Reason for Error	
No vehicle Message QUERY_SUB_STATE	Vehicle id unknown	
Calls		
Function	Where Described	
LOOKUP_VEHICLE	Sec. 2.9.3.2	
generate status report	Sec. 2.14.1.1.11	
ERROR_OUT	Sec. 2.5.2.2	

Table 2.4-175: sbx_connection_query_sub_state CSU [8.4.3.2.38]

2.4.3.2.39 sbx_connection_set_ivis_xmit_modes CSU

This CSU processes the IVIS transmit modes message indicating whether IVIS packets are transmitted to SIMNET and whether IVIS reports are sent to the Symbolics, and adjusts the IVIS transmit modes.

Parameters		
Parameters	Type	Where Typedef Declared
msg_ptr	pointer to IVIS_XMIT_MODES_MSG	Sec. 2.4.1.1
Errors		
Error Name	Reason for Error	
No vehicle Message SET_IVIS_MODES.	Vehicle id unknown	
Calls		
Function	Where Described	
LOOKUP_VEHICLE	Sec. 2.9.3.2	
ERROR_OUT	Sec. 2.5.2.2	

Table 2.4-176: sbx_connection_set_ivis_xmit_mode CSU [8.4.3.2.39]

2.4.3.2.40 sbx_connection_set_ivis_parameters CSU

This CSU processes the IVIS parameters message allowing an operator at the workstation to change key parameters for certain IVIS reports, and adjusts the IVIS parameters.

Parameters		
Parameters	Type	Where Typedef Declared
msg_ptr	pointer to IVIS_PARAMETERS_MSG	Sec. 2.4.1.1
Calls		
Function	Where Described	
square	sim_macros.h	

Table 2.4-177: sbx_connection_set_ivis_parameters CSU [8.4.3.2.40]

2.4.3.2.41 sbx_swap_known_vehicles CSU

This CSU saves what vehicles are known to a workstation.

Parameters		
Parameters	Type	Where Typedef Declared
sbx_connection	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3

Table 2.4-178: sbx_swap_known_vehicles CSU [8.4.3.2.41]

2.4.3.2.42 sbx_set_all_known_vehicles CSU

This CSU creates a connection so that it is aware of about all the vehicles on the battlefield.

Parameters		
Parameters	Type	Where Typedef Declared
sbx_connection	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
Calls		
Function	Where Described	
set_bitfield	Sec. 2.14.3.4.7	

Table 2.4-179: sbx_set_all_known_vehicles CSU [8.4.3.2.42]

2.4.3.2.43 sbx_set_specific_known_vehicles CSU

This CSU creates a connection so that it is aware of only the units that a specified unit can see.

Parameters		
Parameters	Type	Where Typedef Declared
id	unsigned int	Standard
sbx_connection	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
Calls		
Function	Where Described	
sbx_swap_known_vehicles	Sec. 2.4.3.2.41	
clear_bitfield	Sec. 2.14.3.4.6	
LOOKUP_VEHICLE	Sec. 2.9.3.2	
LOOKUP_FORCEID	Sec. 2.9.1.1	
is_enemy	Sec. 2.13.3.1	
set_bit	Sec. 2.14.3.4.3	
or_bitfield	Sec. 2.14.3.4.8	

Table 2.4-180: sbx_set_specific_known_vehicles CSU [8.4.3.2.43]

2.4.3.2.44 sbx_set_top_level_known_vehicles CSU

This CSU creates a connection so that it is aware of only the vehicles that its units can see.

Parameters		
Parameters	Type	Where Typedef Declared
sbx_connection	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3

Calls	
Function	Where Described
sbx_swap_known_vehicles	Sec. 2.4.3.2.41
set_bitfield	Sec. 2.4.3.4.7
clear_bitfield	Sec. 2.14.3.4.6
LOOKUP_VEHICLE	Sec. 2.9.3.2
is_enemy	Sec. 2.13.3.1
LOOKUP_FORCEID	Sec. 2.9.1.1
set_bit	Sec. 2.14.3.4.3
or_bitfield	Sec. 2.14.3.4.8

Table 2.4-181: sbx_set_top_level_known_vehicles CSU [8.4.3.2.44]

2.4.3.2.45 sbx_add_point CSU

This CSU processes a point message, causing a new point to be added to an overlay, or modifies an existing point.

Parameters		
Parameters	Type	Where Typedef Declared
sbx	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
msg_ptr	pointer to POINT_MSG	Sec. 2.4.1.1
Calls		
Function	Where Described	
cm_add_point	Sec. 2.10.2.1.8	

Table 2.4-182: sbx_add_point CSU [8.4.3.2.45]

2.4.3.2.46 sbx_add_area CSU

This CSU processes an area message, causing a new area to be added to an overlay, or modifies an existing area.

Parameters		
Parameters	Type	Where Typedef Declared
sbx	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
msg_ptr	pointer to AREA_MSG	Sec. 2.4.1.1
Calls		
Function	Where Described	
cm_add_area	Sec. 2.10.2.1.9	

Table 2.4-183: sbx_add_area CSU [8.4.3.2.46]

2.4.3.2.47 sbx_add_zone CSU

This CSU processes a zone message, causing a new zone to be added to an overlay, or modifies an existing zone.

Parameters		
Parameters	Type	Where Typedef Declared
sbx	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
msg_ptr	pointer to ZONE_MSG	Sec. 2.4.1.1
Calls		
Function	Where Described	
cm_add_zone	Sec. 2.10.2.1.10	

Table 2.4-184: sbx_add_zone CSU [8.4.3.2.47]

2.4.3.2.48 sbx_add_line CSU

This CSU processes a line message, causing a new line to be added to an overlay, or modifies an existing line.

Parameters		
Parameters	Type	Where Typedef Declared
sbx	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
msg_ptr	pointer to LINE_MSG	Sec. 2.4.1.1
Calls		
Function	Where Described	
cm_add_line	Sec. 2.10.2.1.11	

Table 2.4-185: sbx_add_line CSU [8.4.3.2.48]

2.4.3.2.49 sbx_add_route CSU

This CSU processes a route message, causing a new route to be added to an overlay, or modifies an existing route.

Parameters		
Parameters	Type	Where Typedef Declared
sbx	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
msg_ptr	pointer to ROUTE_MSG	Sec. 2.4.1.1
Calls		
Function	Where Described	
cm_add_route	Sec. 2.10.2.1.12	

Table 2.4-186: sbx_add_route CSU [8.4.3.2.49]

2.4.3.2.50 sbx_delete_overlay CSU

This CSU processes a delete overlay message, causing an overlay to be deleted.

Parameters		
Parameters	Type	Where Typedef Declared
sbx	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
msg_ptr	pointer to DELETE_OVERLAY_MSG	Sec. 2.4.1.1
Calls		
Function	Where Described	
cm_delete_overlay	Sec. 2.10.2.1.13	

Table 2.4-187: sbx_delete_overlay CSU [8.4.3.2.50]

2.4.3.2.51 sbx_execute_overlay CSU

This CSU processes an execute overlay message, causing a unit to follow the indicated route and obey the control measures in the specified overlay.

Parameters		
Parameters	Type	Where Typedef Declared
sbx	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
msg_ptr	pointer to EXECUTE_OVERLAY_MSG	Sec. 2.4.1.1
Calls		
Function	Where Described	
cm_execute_overlay	Sec. 2.10.2.1.20	

Table 2.4-188: sbx_execute_overlay CSU [8.4.3.2.51]

2.4.3.2.52 sbx_halt CSU

This CSU processes a halt message, causing a vehicle or unit to halt.

Parameters		
Parameters	Type	Where Typedef Declared
sbx	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
msg_ptr	pointer to HALT_MSG	Sec. 2.4.1.1

Calls	
Function	Where Described
LOOKUP_VEHICLE	Sec. 2.9.3.2
saf_vehicle_halt	Sec. 2.6.1.1.52
composite_halt	Sec. 2.8.1.3.43

Table 2.4-189: sbx_halt CSU [8.4.3.2.52]

2.4.3.2.53 sbx_change_speed CSU

This CSU processes a change speed message, causing a vehicle or unit to change speed.

Parameters		
Parameters	Type	Where Typedef Declared
sbx	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
msg_ptr	pointer to CHANGE_SPEED_MSG	Sec. 2.4.1.1
Calls		
Function	Where Described	
LOOKUP_VEHICLE	Sec. 2.9.3.2	
saf_vehicle_change_speed	Sec. 2.6.1.1.53	
composite_change_speed	Sec. 2.8.1.3.44	

Table 2.4-190: sbx_change_speed CSU [8.4.3.2.53]

2.4.3.2.54 sbx_change_formation CSU

This CSU processes a change formation message, causing a change in formation.

Parameters		
Parameters	Type	Where Typedef Declared
sbx	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
msg_ptr	pointer to CHANGE_FORMATION_MSG	Sec. 2.4.1.1
Calls		
Function	Where Described	
LOOKUP_VEHICLE	Sec. 2.9.3.2	
composite_change_formation	Sec. 2.8.1.3.47	

Table 2.4-191: sbx_change_formation CSU [8.4.3.2.54]

2.4.3.2.55 sbx_delete_cm CSU

This CSU processes a delete control measure message, causing a control measure to be deleted from an overlay.

Parameters		
Parameters	Type	Where Typedef Declared
sbx	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
msg_ptr	pointer to DELETE_CM_MSG	Sec. 2.4.1.1
Calls		
Function	Where Described	
cm_delete_cm	Sec. 2.10.2.1.14	

Table 2.4-192: sbx_delete_cm CSU [8.4.3.2.55]

2.4.3.2.56 sbx_rejoin_unit CSU

This CSU processes a rejoin unit message, causing a vehicle or unit to abandon its current mission, as assigned by an execute overlay message, and rejoin its superior unit (if a superior unit exists).

Parameters		
Parameters	Type	Where Typedef Declared
sbx	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
msg_ptr	pointer to REJOIN_UNIT_MSG	Sec. 2.4.1.1
Calls		
Function	Where Described	
LOOKUP_VEHICLE	Sec. 2.9.3.2	
saf_vehicle_rejoin_unit	Sec. 2.6.1 1.45	
composite_rejoin_unit	Sec. 2.8.1 3.39	

Table 2.4-193: sbx_rejoin_unit CSU [8.4.3.2.56]

2.4.3.2.57 sbx_follow_vehicle CSU

This CSU processes a follow vehicle message, causing a vehicle or unit to follow a vehicle.

Parameters		
Parameters	Type	Where Typedef Declared
sbx	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
msg_ptr	pointer to FOLLOW_VEHICLE_MSG	Sec. 2.4.1.1

Calls	
Function	Where Described
LOOKUP_VEHICLE	Sec. 2.9.3.2
saf_vehicle_follow_vehicle	Sec. 2.6.1.1.54
composite_follow_vehicle	Sec. 2.8.1.3.45

Table 2.4-194: sbx_follow_vehicle CSU [8.4.3.2.57]

2.4.3.2.58 sbx_simulator_in_command CSU

This CSU processes a simulator in command message, causing the leading vehicle of a unit to be deactivated. The rest of the unit then follows the specified vehicle, which must be a remote vehicle.

Parameters		
Parameters	Type	Where Typedef Declared
sbx	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
msg_ptr	pointer to SIMULATOR_IN_COMMAND_MSG	Sec. 2.4.1.1
Calls		
Function	Where Described	
LOOKUP_SAFOBJ	Sec. 2.9.1.1	
sbx_printf	Sec. 2.4.3.2.8	
LOOKUP_VEHICLE	Sec. 2.9.3.2	
composite_simulator_in_command	Sec. 2.8.1.3.46	

Table 2.4-195: sbx_simulator_in_command CSU [8.4.3.2.58]

2.4.3.2.59 sbx_goto_point CSU

This CSU processes a go to point message, causing a unit or vehicle to go to a specified point.

Parameters		
Parameters	Type	Where Typedef Declared
sbx	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
msg_ptr	pointer to GOTO_POINT_MSG	Sec. 2.4.1.1

Calls	
Function	Where Described
string for hold type	Sec. 2.4.3.2.66
LOOKUP VEHICLE	Sec. 2.9.3.2
saf vehicle goto point	Sec. 2.6.1.1.55
composite goto point	Sec. 2.8.1.3.48

Table 2.4-196: sbx_goto_point CSU [8.4.3.2.59]

2.4.3.2.60 sbx_resume_mission CSU

This CSU processes a resume mission message, causing a unit or vehicle to return to the current mission after being interrupted by an immediate intervention.

Parameters		
Parameters	Type	Where Typedef Declared
sbx	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
msg_ptr	pointer to RESUME_MISSION_MSG	Sec. 2.4.1.1
Calls		
Function	Where Described	
LOOKUP VEHICLE	Sec. 2.9.3.2	
saf vehicle resume mission	Sec. 2.6.1.1.56	
composite resume mission	Sec. 2.8.1.3.49	

Table 2.4-197: sbx_resume_mission CSU [8.4.3.2.60]

2.4.3.2.61 sbx_face_direction CSU

This CSU processes a face direction message, causing a unit or vehicle to face a specified direction.

Parameters		
Parameters	Type	Where Typedef Declared
sbx	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
msg_ptr	pointer to FACE_DIRECTION_MSG	Sec. 2.4.1.1
Calls		
Function	Where Described	
LOOKUP VEHICLE	Sec. 2.9.3.2	
saf vehicle face direction	Sec. 2.6.1.1.57	
composite face direction	Sec. 2.8.1.3.50	

Table 2.4-198: sbx_face_direction CSU [8.4.3.2.61]

2.4.3.2.62 sbx_set_targeting_parameters CSU

This CSU processes a set targeting message, causing the targeting parameters to be set.

Parameters		
Parameters	Type	Where Typedef Declared
sbx	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
msg_ptr	pointer to SET_TARGETING_MSG	Sec. 2.4.1.1
Calls		
Function	Where Described	
firestatus to string	Sec. 2.6.9.3.4	
set_targeting_parameters	Sec. 2.14.1.1.7	

Table 2.4-199: sbx_set_targeting_parameters CSU [8.4.3.2.62]

2.4.3.2.63 sbx_land CSU

This CSU processes a land message, causing an air-unit or vehicle to land.

Parameters		
Parameters	Type	Where Typedef Declared
sbx	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
msg_ptr	pointer to LAND_MSG	Sec. 2.4.1.1
Calls		
Function	Where Described	
LOOKUP_VEHICLE	Sec. 2.9.3.2	
composite_land	Sec. 2.8.1.3.52	
pilot_land_im	Sec. 2.6.4.2.70	

Table 2.4-200: sbx_land CSU [8.4.3.2.63]

2.4.3.2.64 string_for_altitude_type CSU

This CSU returns a human-readable string identifying an altitude type.

Parameters		
Parameters	Type	Where Typedef Declared
type	int	Standard
ReturnValues		
Return Value	Type	Meaning
"abs"	char	Absolute altitude
"agl"	char	Altitude above ground level
"unknown type"	char	Unknown altitude type

Table 2.4-201: string_for_altitude_type CSU [8.4.3.2.64]

2.4.3.2.65 sbx_altitude CSU

This CSU processes an altitude message, causing an air-unit or vehicle to change altitude.

Parameters		
Parameters	Type	Where Typedef Declared
sbx	pointer to SBX CONNECTION VARS	Sec. 2.4.3.3
msg_ptr	pointer to ALTITUDE MSG	Sec. 2.4.1.1
Calls		
Function	Where Described	
string for altitude type	Sec. 2.4.3.2.64	
LOOKUP VEHICLE	Sec. 2.9.3.2	
composite change altitude	Sec. 2.8.1.3.53	
pilot change altitude im	Sec. 2.6.4.2.66	

Table 2.4-202: sbx_altitude CSU [8.4.3.2.65]

2.4.3.2.66 string_for_hold_type CSU

This CSU returns a human-readable string identifying a hold type.

Parameters		
Parameters	Type	Where Typedef Declared
type	int	Standard
ReturnValues		
Return Value	Type	Meaning
"hover"	char	Hover hold
"orbit"	char	Orbit hold
"race track"	char	Race track hold
"unknown type"	char	Unknown hold type

Table 2.4-203: string_for_hold_type CSU [8.4.3.2.66]

2.4.3.2.67 sbx_hold CSU

This CSU processes a hold message, causing an air-unit or vehicle to hold.

Parameters		
Parameters	Type	Where Typedef Declared
sbx	pointer to SBX CONNECTION VARS	Sec. 2.4.3.3
msg_ptr	pointer to HOLD MSG	Sec. 2.4.1.1

Calls	
Function	Where Described
string for hold type	Sec. 2.4.3.2.66
LOOKUP VEHICLE	Sec. 2.9.3.2
composite halt	Sec. 2.8.1.3.43
pilot_hold_im	Sec. 2.6.4.2.76

Table 2.4-204: sbx_hold CSU [8.4.3.2.67]

2.4.3.2.68 string_for_attack_type CSU

This CSU returns a human-readable string identifying an attack type.

Parameters		
Parameters	Type	Where Typedef Declared
type	int	Standard
ReturnValues		
Return Value	Type	Meaning
"running fire attack"	char	Running fire attack
"popup attack"	char	Popup attack
"unknown type"	char	Unknown attack type

Table 2.4-205: string_for_attack_type CSU [8.4.3.2.68]

2.4.3.2.69 sbx_attack CSU

This CSU processes an attack message, causing an air-unit or vehicle to attack.

Parameters		
Parameters	Type	Where Typedef Declared
sbx	pointer to SBX_CONNECTION_VARS	Sec. 2.4.3.3
msg_ptr	pointer to ATTACK_MSG	Sec. 2.4.1.1
Calls		
Function	Where Described	
string for attack type	Sec. 2.4.3.2.68	
LOOKUP VEHICLE	Sec. 2.9.3.2	
composite attack	Sec. 2.8.1.3.51	
pilot_hoverattack im	Sec. 2.6.4.2.75	

Table 2.4-206: sbx_attack CSU [8.4.3.2.69]

2.4.3.2.70 sbx_attach_stealth CSU

This CSU handles an attach stealth message, causing a stealth to attach to a unit or vehicle.

Parameters		
Parameters	Type	Where Typedef Declared
sbx	pointer to SBX CONNECTION VARS	Sec. 2.4.3.3
msg_ptr	pointer to ATTACH STEALTH MSG	Sec. 2.4.1.1
Calls		
Function	Where Described	
LOOKUP_VEHICLE	Sec. 2.9.3.2	
SAF_COMPOSITE_P		
composite_find_vehicle_for_stealth	Sec. 2.8.1.3.29	
FOR_VEHICLES_DO	Sec. 2.9.3.2	
simnet_id_from_saf_id		
vec_copy	Sec. 2.14.3.5.17	
OBJ_POSITION	Sec. 2.9.1.1	
stealth_send_teleport_to		
stealth_send_attach_to_vehicle		
OBJ_VEHICLEID	Sec. 2.9.1.1	
remote_change_stealth_controlling_port	Sec. 2.7.1.16	

Table 2.4-207: sbx_attach_stealth CSU [8.4.3.2.70]

2.4.3.2.71 sbx_resupply CSU

This CSU processes a resupply message, causing a vehicle to resupply from a resupply vehicle.

Parameters		
Parameters	Type	Where Typedef Declared
sbx	pointer to SBX CONNECTION VARS	Sec. 2.4.3.3
msg_ptr	pointer to RESUPPLY MSG	Sec. 2.4.1.1
Calls		
Function	Where Described	
LOOKUP_VEHICLE	Sec. 2.9.3.2	
start_resupply_of_to	Sec. 2.6.6.1.4	

Table 2.4-208: sbx_resupply CSU [8.4.3.2.71]

2.4.3.2.72 broadcast_echelon_data CSU

This CSU causes a unit to send an echelon packet to all connections.

Parameters		
Parameters	Type	Where Typedef Declared
on_whom	unsigned int	Standard
Errors		
Error Name	Reason for Error	
Cannot send status of nonexistent vehicle	Vehicle does not exist	
Calls		
Function	Where Described	
LOOKUP_VEHICLE	Sec. 2.9.3.2	
ERROR_OUT	Sec. 2.5.2.2	
buffer_allocate	Sec. 2.14.4.1.10	
fill_in echelon data	Sec. 2.14.1.1.2	
fill_sbx_opfor_header	Sec. 2.4.3.2.18	
write_buffer_all_sbx	Sec. 2.4.3.2.3	
buffer_deallocate	Sec. 2.14.4.2.15	

Table 2.4-209: broadcast_echelon_data CSU [8.4.3.2.72]

2.4.3.2.73 broadcast_appearance_data CSU

This CSU causes a unit to send an appearance packet to all connections.

Parameters		
Parameters	Type	Where Typedef Declared
on_whom	unsigned int	Standard
Errors		
Error Name	Reason for Error	
Cannot send status of nonexistant vehicle	Vehicle does not exist	
Calls		
Function	Where Described	
LOOKUP VEHICLE	Sec. 2.9.3.2	
ERROR OUT	Sec. 2.5.2.2	
buffer allocate	Sec. 2.14.4.1.10	
fill in appearance data	Sec. 2.14.1.1.1	
fill sbx opfor header	Sec. 2.4.3.2.18	
write buffer all sbx	Sec. 2.4.3.2.3	
buffer deallocate	Sec. 2.14.4.2.15	

Table 2.4-210: broadcast_appearance_data CSU [8.4.3.2.73]

2.4.3.2.74 broadcast_pae_data CSU

This CSU causes a unit to send a PAE packet to all connections.

Parameters		
Parameters on whom	Type	Where Typedef Declared
	unsigned int	Standard
Errors		
Error Name	Reason for Error	
Cannot send status of nonexistent vehicle	Vehicle does not exist	
Calls		
Function	Where Described	
LOOKUP_VEHICLE	Sec. 2.9.3.2	
ERROR_OUT	Sec. 2.5.2.2	
buffer_allocate	Sec. 2.14.4.1.10	
fill_in_position_data	Sec. 2.14.1.1.3	
fill_in_appearance_data	Sec. 2.14.1.1.1	
fill_in_echelon_data	Sec. 2.14.1.1.2	
fill_sbx_opfor_header	Sec. 2.4.3.2.18	
write_buffer_all_sbx	Sec. 2.4.3.2.3	
buffer_deallocate	Sec. 2.14.4.2.15	

Table 2.4-211: broadcast_pae_data CSU [8.4.3.2.74]

2.4.3.2.75 broadcast_veh_is_gone CSU

This CSU causes a unit to send a vehicle-is-gone packet to all connections.

Parameters		
Parameters	Type	Where Typedef Declared
vehicle id	unsigned int	Standard
Calls		
Function	Where Described	
buffer allocate	Sec. 2.14.4.2.12	
fill sbx opfor header	Sec. 2.4.3.2.18	
write buffer all sbx	Sec. 2.4.3.2.3	
buffer deallocate	Sec. 2.14.4.2.15	

Table 2.4-212: broadcast_veh_is_gone CSU [8.4.3.2.75]

2.4.3.2.76 print_opfor_header CSU

This CSU prints the fields in an opfor header.

Parameters		
Parameters	Type	Where Typedef Declared
hdr_ptr	pointer to OPFOR HEADER	Sec. 2.4.1.1

Table 2.4-213: print_opfor_header CSU [8.4.3.2.76]

2.4.3.2.77 print_message_position CSU

This CSU prints the fields in a vector.

Parameters		
Parameters	Type	Where Typedef Declared
vec_ptr	pointer to REAL	sim-types.h

Table 2.4-214: print_message_position CSU [8.4.3.2.77]

2.4.3.2.78 print_vehicle_ids CSU

This CSU prints zero or more vehicle ids.

Parameters		
Parameters	Type	Where Typedef Declared
id_ptr	pointer to unsigned int	Standard
count	int	Standard

Table 2.4-215: print_vehicle_ids CSU [8.4.3.2.78]

2.4.3.2.79 sbx_connection_show_top_level_units CSU

This CSU causes the top level units to be printed.

Parameters		
Parameters	Type	Where Typedef Declared
sbx_connection	pointer to SBX CONNECTION VARS	Sec. 2.4.3.3

Table 2.4-216: sbx_connection_show_top_level_units CSU [8.4.3.2.79]

2.4.3.3 sbx.h CSU

/simnet/src/host/sbx.h

This CSU contains the structure definition (SBX_CONNECTION_VARS) and symbolic constants used by the workstation communication code, both shown below. In addition, the file contains the macro definition for allocate_sbx_connection contained in Appendix A.

Item	Type	Where Type Defined
stat_iterator	VEHICLE_ITERATOR	Sec. 2.9.3.2
s	pointer to BUF_RUDP_SOCKET	Sec. 2.4.2.6
wat_pulse_event	int	Standard
stat_pulse_event	int	Standard
wat_index	int	Standard
port_number	int	Standard
forceID	ForceID	basic.h
both_forces	int	Standard
num_top_level_units	unsigned short	Standard
top_level_units	unsigned int	Standard
vehicles_known_to_workstation	pointer to char	Standard
last_vehicles_known_to_workstation	pointer to char	Standard
last_poll_mode	int	Standard
overlays	pointer to OVERLAY_LIST	Sec. 2.10.2.2
stealth_address	SimulationAddress	address.h

Table 2.4-217: SBX_CONNECTION_VARS Structure Definition

Constant	Value
MAX_TOP_LEVEL_UNITS	32
MAX_SBX_CONNS	16

Table 2.4-218: sbx.h Symbolic Constants

2.5 PARSER INTERFACE CSC

The parser interface code handles command inputs from the STDIN device of the UNIX process which is running the simulation. This device can be the SAF Simhost HWCI console (a WYSE60 RS-232 terminal) or a remote login from a terminal emulator on some other machine such as the SAF Workstation HWCI.

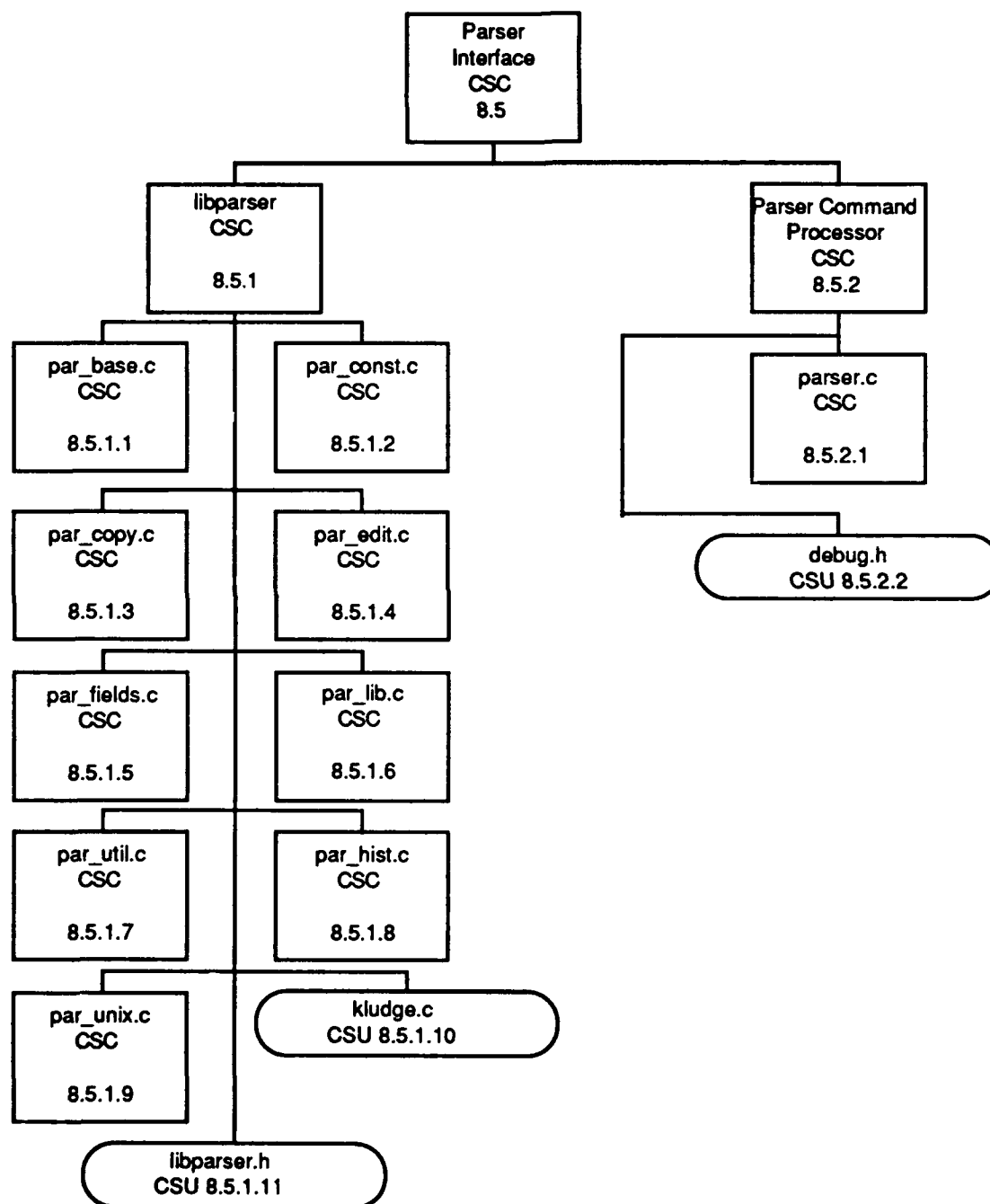


Figure 2.5-1: Parser Interface CSC Structure

The parser commands perform system operator functions, such as controlling the amount of debugging output printed at the STDIN device, and inspecting the state of various simulation objects, network statistics, and simulation loading. When ticked by the scheduler, the parser code processes its input buffer and returns output. The simulation is stopped by an exit command from the parser.

2.5.1 libparser CSC

/simnet/libsrc/libparser

This library does the work of conversion for the parser inputs code.

2.5.1.1 par_base.c CSC

/simnet/libsrc/libparser/par_base.c

The CSUs in this CSC are those parse functions that have special knowledge; therefore, they can not be added like user-addable parse functions.

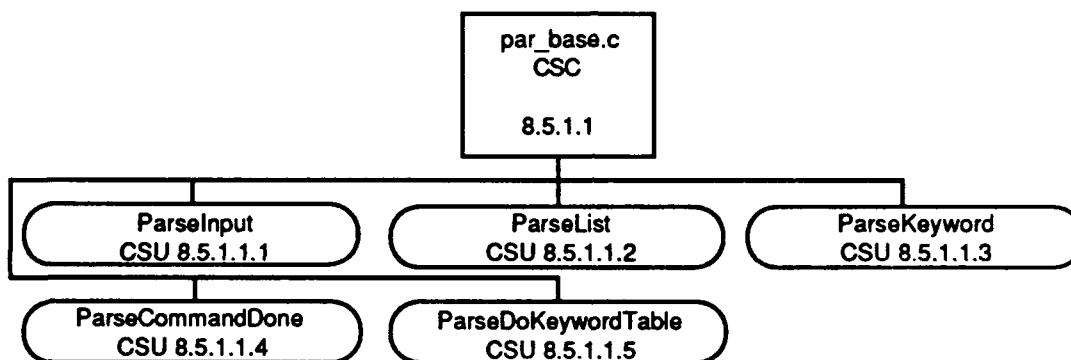


Figure 2.5-2: par_base.c CSC Structure

2.5.1.1.1 ParseInput CSU

This function executes a parse table. It finds the end of the table and calls ParseList to do the work. It then processes the parse value returned by ParseList.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
table	pointer to register PARSE_TABLE	Sec. 2.5.1.11

Calls	
Function	Where Described
BeginningOfLine	Sec. 2.5.1.4.11
ParseList	Sec. 2.5.1.1.2
RestoreCursor	Sec. 2.5.1.4.9
KillForward	Sec. 2.5.1.4.14
ClearLine	Sec. 2.5.1.4.19
EmptyLine	Sec. 2.5.1.4.10
ParseError	Sec. 2.5.1.7.7
EndOfLine	Sec. 2.5.1.4.14
BackChar	Sec. 2.5.1.4.25
NextChar	Sec. 2.5.1.4.7
InsertChar	Sec. 2.5.1.4.23
Free	Sec. 2.5.1.9.2
UpdateLine	Sec. 2.5.1.4.3
RedisplayLine	Sec. 2.5.1.4.5

Table 2.5-1: ParseInput CSU [8.5.1.1.1]

2.5.1.1.2 ParseList CSU

This CSU searches through the parse table, reading the table token by token and executing all parse functions associated with non-keyword functionality (e.g., loading data).

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
argc	int	Standard
first	pointer to register PARSE_TABLE	Sec. 2.5.1.11
last	pointer to register PARSE_TABLE	Sec. 2.5.1.11
ReturnValues		
Return Value	Type	Meaning
PARSE_ERROR	int	Extra data on line
PARSE_SUCCESS	int	Successful parse
PARSE_IGNORE	int	Needs completed command
val	int	Returned new value due to illegal command if processing P_PARSE_FUNCTION and val
argc	int	new value due to illegal command

Errors	
Error Name	Reason for Error
PARSE_ERROR	Extra data on line
PARSE_IGNORE	Command incomplete
val	Returned new value due to illegal command if processing P_PARSE_FUNCTION and val
argc	Returned new value due to illegal command
Calls	
Function	Where Described
ParsePrint	Sec. 2.5.1.9.3
ParseGetToken	Sec. 2.5.1.7.6
ParseError	Sec. 2.5.1.7.7
CommandLog	Sec. 2.5.1.8.1
ParseMessage	Sec. 2.5.1.7.8
ParseFindEndList	Sec. 2.5.1.7.9
ParseKeyword	Sec. 2.5.1.1.3
TypeFields	Sec. 2.5.1.5.2

Table 2.5-2: ParseList CSU [8.5.1.1.2]

2.5.1.1.3 ParseKeyword CSU

This CSU processes a keyword list by determining the next keyword and processing the commands that follow it.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
argc	int	Standard
argv	pointer to char	Standard
first	pointer to register PARSE_TABLE	Sec. 2.5.1.11
end	pointer to register PARSE_TABLE	Sec. 2.5.1.11
ReturnValues		
Return Value	Type	Meaning
PARSE_NEEDKEYWORD	int	out of tokens
PARSE_ERROR	int	error has occurred
PARSE_IGNORE	int	ignore status
PARSE_UPDATE_LINE	int	update line
(ParseList())	int	new values of argc
argc	int	new value of argc
ERROR	int	error occurred

Errors	
Error Name	Reason for Error
PARSE ERROR	No keyword match
PARSE IGNORE	Other parse table token error (ie. not unique keyword)
ERROR	Internal error in parser
Calls	
Function	Where Described
ParsePrint	Sec. 2.5.1.9.3
ParseGetToken	Sec. 2.5.1.7.6
ParseMatch	Sec. 2.5.1.7.10
ParseFindEndList	Sec. 2.5.1.7.9
ParseError	Sec. 2.5.1.7.7
ParseEscapeComplete	Sec. 2.5.1.7.12
ParseList	Sec. 2.5.1.1.2

Table 2.5-3: ParseKeyword CSU [8.5.1.1.3]

2.5.1.1.4 ParseCommandDone CSU

This CSU terminates a command list.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE DATA	Sec. 2.5.1.11
nu argc	int	Standard
nu argv[]	pointer to char	Standard
arg2	pointer to char	Standard
ReturnValues		
Return Value	Type	Meaning
PARSE ERROR	int	Extra data in line
PARSE SUCCESS	int	Parse successful
PARSE IGNORE	int	No carriage return at line end
Calls		
Function	Where Described	
ParseGetToken	Sec. 2.5.1.7.6	
ParseError	Sec. 2.5.1.7.7	
ParseMessage	Sec. 2.5.1.7.8	

Table 2.5-4: ParseCommandDone CSU [8.5.1.1.4]

2.5.1.1.5 ParseDoKeywordTable CSU

This CSU executes a new keyword table.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
argc	int	Standard
argv[]	pointer to char	Standard
arg2	pointer to char	Standard
ReturnValues		
Return Value	Type	Meaning
(ParseList())	int	new value for argc
Calls		
Function	Where Described	
ParseList	Sec. 2.5.1.1.2	

Table 2.5-5: ParseDoKeywordTable CSU [8.5.1.1.5]

2.5.1.2 par_const.c CSC

/simnet/libsrc/libparser/par_const.c

This file contains the CSU used to parse constant tables.

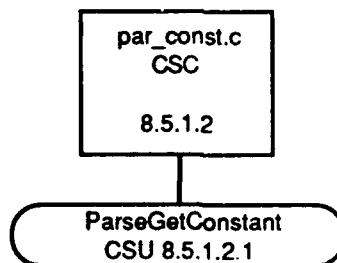


Figure 2.5-3: par_const.c CSC Structure

2.5.1.2.1 ParseGetConstant CSU

The purpose of this CSU is to obtain a constant value.

Parameters		
Parameters	Type	Where Typedef Declared
ptr	pointer to register PARSE_DATA	Sec. 2.5.1.11
argc	int	Standard
argv[]	pointer to char	Standard
ftable	pointer to register PARSE_TABLE	Sec. 2.5.1.11

ReturnValues		
Return Value	Type	Meaning
PARSE IGNORE	int	
PARSE ERROR	int	
matched	int	
argc+1	int	
PARSE UPDATE LINE	int	
PARSE ADDSPACE	int	
argc	int	
Calls		
Function	Where Described	
ParseGetToken	Sec. 2.5.1.7.6	
ParseError	Sec. 2.5.1.7.7	
ParseTableFind	Sec. 2.5.1.7.11	
ParseEscapeComplete	Sec. 2.5.1.7.12	
p_arg0	Sec. 2.5.1.11 See Appendix A	
ParseMatch	Sec. 2.5.1.7.10	
ParsePrint	Sec. 2.5.1.9.3	

Table 2.5-6: ParseGetConstant CSU [8.5.1.2.1]

2.5.1.3 par_copy.c CSC

/simnet/libsrc/libparser/par_copy.c

This file contains one CSU, stringcopy.

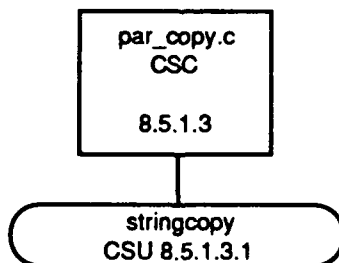


Figure 2.5-4: par_copy.c CSC Structure

2.5.1.3.1 stringcopy CSU

This CSU copies a string of characters.

Parameters		
Parameters	Type	Where Typedef Declared
str	pointer to register char	Standard
ReturnValues		
Return Value	Type	Meaning
p	char	a string that was copied

Calls	
Function	Where Described
Alloc	Sec. 2.5.1.9.1

Table 2.5-7: stringcopy CSU [8.5.1.3.1]

2.5.1.4 par_edit.c CSC

/simnet/libsrc/libparser/par_edit.c

This CSC contains the CSUs for the file editing commands.

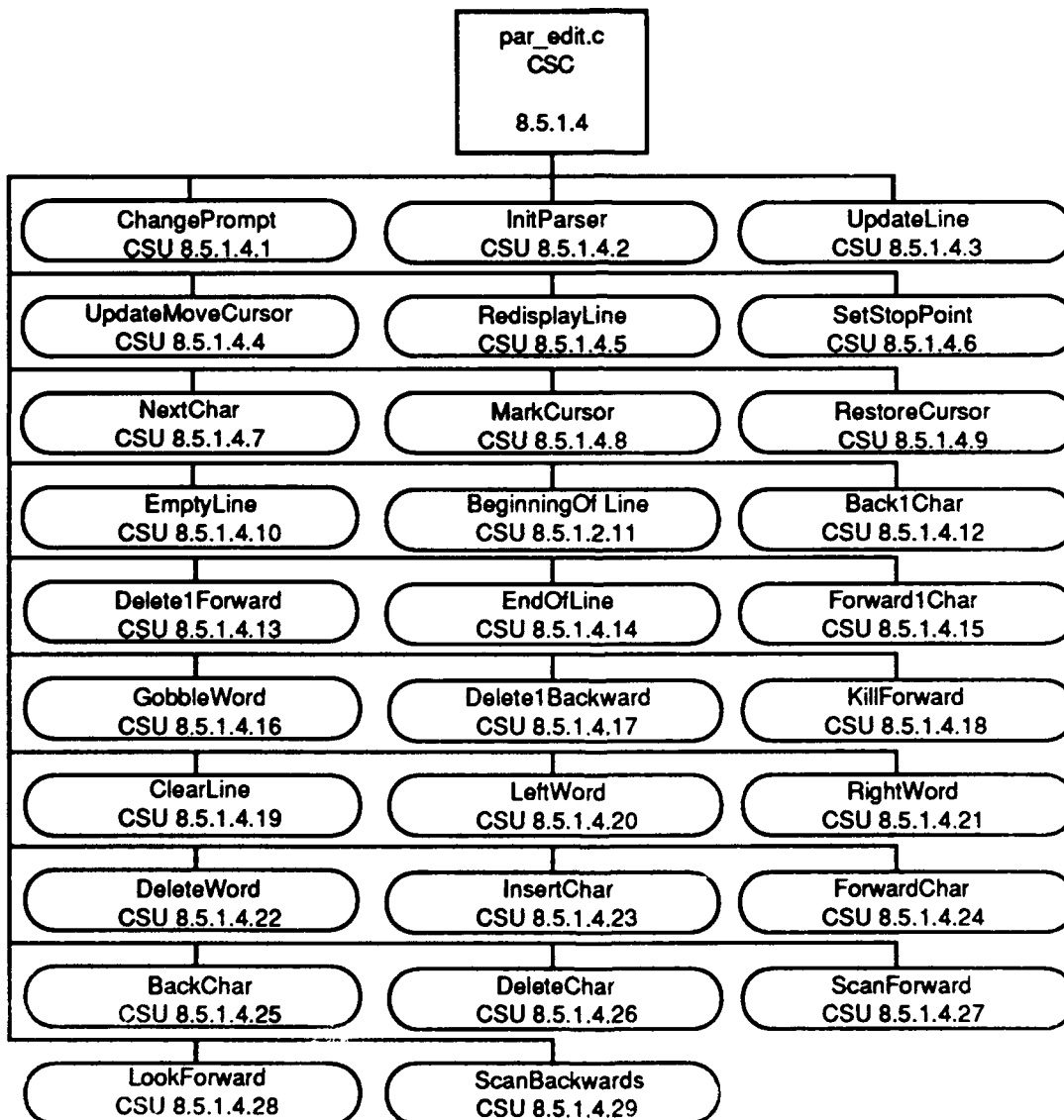


Figure 2.5-5. par_edit.c CSC Structure

2.5.1.4.1 ChangePrompt CSU

This CSU resets the parser prompt.

Parameters		
Parameters	Type	Where Typedef Declared
prompt	pointer to char	Standard

Table 2.5-8: ChangePrompt CSU [8.5.1.4.1]

2.5.1.4.2 InitParser CSU

This CSU initializes the parser.

Parameters		
Parameters	Type	Where Typedef Declared
table	pointer to PARSE_TABLE	Sec. 2.5.1.11
prompt	pointer to char	Standard
logsize	int	Standard
ReturnValues		
Return Value	Type	Meaning
pdp	pointer to PARSE_DATA	a pointer to the parser
Calls		
Function	Where Described	
Alloc	Sec. 2.5.1.9.1	
ParsePrint	Sec. 2.5.1.9.3	

Table 2.5-9: InitParser CSU [8.5.1.4.2]

2.5.1.4.3 UpdateLine CSU

This CSU is the "heart" of the editor, updating what is on the screen to the contents of the internal input buffer. This version of the CSU assumes that it is being run on a display capable of backspaces.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
Calls		
Function	Where Described	
UpdateMoveCursor	Sec. 2.5.1.4.4	
ParsePrint	Sec. 2.5.1.9.3	

Table 2.5-10: UpdateLine CSU [8.5.1.4.3]

2.5.1.4.4 UpdateMoveCursor CSU

This CSU updates the position of the cursor.

Parameters		
Parameters	Type	Where Typedef Declared
outbuf	pointer to register char	Standard
linebuf	pointer to register char	Standard
count	register int	Standard
ReturnValues		
Return Value	Type	Meaning
outbuf	pointer to char	oints to the position of the cursor

Table 2.5-11: UpdateMoveCursor CSU [8.5.1.4.4]

2.5.1.4.5 RedisplayLine CSU

This CSU redisplay the current line, including the user prompt. The calls to lock_wait and lock_unlock are made only if a multiprocessor system is being used (i.e., if MUTIPROCESSOR is defined).

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
Calls		
Function	Where Described	
lock_wait		
ParsePrint	Sec. 2.5.1.9.3	
UpdateLine	Sec. 2.5.1.4.3	
lock_unlock		

Table 2.5-12: RedisplayLine CSU [8.5.1.4.5]

2.5.1.4.6 SetStopPoint CSU

This CSU informs NextChar, Sec. 2.5.1.4.7, where to stop reading.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
at_cursor	int	Standard

Table 2.5-13: SetStopPoint CSU [8.5.1.4.6]

2.5.1.4.7 NextChar CSU

This CSU returns the next character from the input buffer, updating the buffer cursor, if it is not at the stop point. If it is at the stop point, it returns '\0'.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
ReturnValues		
Return Value	Type	Meaning
'\0'	char	End of line
(*pdp->Cur++)	char	Next character

Table 2.5-14: NextChar CSU [8.5.1.4.7]

2.5.1.4.8 MarkCursor CSU

This CSU marks the current cursor position for later restoration.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11

Table 2.5-15: MarkCursor CSU [8.5.1.4.8]

2.5.1.4.9 RestoreCursor CSU

This CSU restores the cursor to the last cursor mark.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11

Table 2.5-16: RestoreCursor CSU [8.5.1.4.9]

2.5.1.4.10 EmptyLine CSU

This CSU determines if the input line is empty.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11

ReturnValues		
Return Value	Type	Meaning
TRUE	int	Input line is empty
FALSE	int	Input line is not empty

Table 2.5-17: EmptyLine CSU [8.5.1.4.10]

2.5.1.4.11 BeginningOfLine CSU

This CSU places the cursor at the beginning of the line.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11

Table 2.5-18: BeginningOfLine CSU [8.5.1.4.11]

2.5.1.4.12 Back1Char CSU

This CSU moves the cursor back one character.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
Calls		
Function	Where Described	
BackChar	Sec. 2.5.1.4.25	

Table 2.5-19: Back1Char CSU [8.5.1.4.12]

2.5.1.4.13 Delete1Forward CSU

This CSU deletes the next character in the line.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
Calls		
Function	Where Described	
DeleteChar	Sec. 2.5.1.4.26	

Table 2.5-20: Delete1Forward CSU [8.5.1.4.13]

2.5.1.4.14 EndOfLine CSU

This CSU moves the cursor to the end of the line.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE DATA	Sec. 2.5.1.11

Table 2.5-21: EndOfLine CSU [8.5.1.4.14]

2.5.1.4.15 Forward1Char CSU

This CSU moves the cursor forward one character.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE DATA	Sec. 2.5.1.11
Calls		
Function	Where Described	
ForwardChar	Sec. 2.5.1.4.24	

Table 2.5-22: Forward1Char CSU [8.5.1.4.15]

2.5.1.4.16 GobbleWord CSU

This CSU deletes one word forward.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE DATA	Sec. 2.5.1.11
Calls		
Function	Where Described	
LookForward	Sec. 2.5.1.4.28	
DeleteChar	Sec. 2.5.1.4.26	

Table 2.5-23: GobbleWord CSU [8.5.1.4.16]

2.5.1.4.17 Delete1Backward CSU

This CSU deletes the last character.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE DATA	Sec. 2.5.1.11

Calls	
Function	Where Described
BackChar	Sec. 2.5.1.4.25
DeleteChar	Sec. 2.5.1.4.26

Table 2.5-24: Delete1Backward CSU [8.5.1.4.17]

2.5.1.4.18 KillForward CSU

This CSU deletes the characters from the cursor position to the end of the line.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE DATA	Sec. 2.5.1.11
Calls		
Function	Where Described	
DeleteChar	Sec. 2.5.1.4.26	

Table 2.5-25: KillForward CSU [8.5.1.4.18]

2.5.1.4.19 ClearLine CSU

This CSU is used after successfully parsing a line. It clears pdp.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE DATA	Sec. 2.5.1.11

Table 2.5-26: ClearLine CSU [8.5.1.4.19]

2.5.1.4.20 LeftWord CSU

This CSU moves the cursor one word to the left.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE DATA	Sec. 2.5.1.11
Calls		
Function	Where Described	
ScanBackwards	Sec. 2.5.1.4.29	
BackChar	Sec. 2.5.1.4.25	

Table 2.5-27: LeftWord CSU [8.5.1.4.20]

2.5.1.4.21 RightWord CSU

This CSU moves the cursor one word to the right.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
Calls		
Function	Where Described	
ScanForward	Sec. 2.5.1.4.27	
ForwardChar	Sec. 2.5.1.4.24	

Table 2.5-28: RightWord CSU [8.5.1.4.21]

2.5.1.4.22 DeleteWord CSU

This CSU deletes the last word.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
Calls		
Function	Where Described	
ScanBackwards	Sec. 2.5.1.4.29	
BackChar	Sec. 2.5.1.4.25	
DeleteChar	Sec. 2.5.1.4.26	

Table 2.5-29: DeleteWord CSU [8.5.1.4.22]

2.5.1.4.23 InsertChar CSU

This CSU inserts a character at the current position.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
ch	register char	Standard
Calls		
Function	Where Described	
ParsePrint	Sec. 2.5.1.9.3	
InsertChar	Sec. 2.5.1.4.23	
BackChar	Sec. 2.5.1.4.25	

Table 2.5-30: InsertChar CSU [8.5.1.4.23]

2.5.1.4.24 ForwardChar CSU

This CSU moves forward a specified number of characters, but not beyond the end of the string.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE DATA	Sec. 2.5.1.11
count	register int	Standard
Calls		
Function	Where Described	
ParsePrint	Sec. 2.5.1.9.3	

Table 2.5-31: ForwardChar CSU [8.5.1.4.24]

2.5.1.4.25 BackChar CSU

This CSU moves back a specified number of characters, but not beyond the first one.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE DATA	Sec. 2.5.1.11
count	register int	Standard

Table 2.5-32: BackChar CSU [8.5.1.4.25]

2.5.1.4.26 DeleteChar CSU

This CSU deletes a specified number characters.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE DATA	Sec. 2.5.1.11
count	register int	Standard
Calls		
Function	Where Described	
BackChar	Sec. 2.5.1.4.25	

Table 2.5-33: DeleteChar CSU [8.5.1.4.26]

2.5.1.4.27 ScanForward CSU

This CSU returns a pointer to the next word.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
ReturnValues		
Return Value	Type	Meaning
p	pointer to char	New cursor point

Table 2.5-34: ScanForward CSU [8.5.1.4.27]

2.5.1.4.28 LookForward CSU

This CSU returns a pointer to the next word or whitespace.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
ReturnValues		
Return Value	Type	Meaning
*p	pointer to char	Pointer to new cursor

Table 2.5-35: LookForward CSU [8.5.1.4.28]

2.5.1.4.29 ScanBackwards CSU

This CSU returns a pointer to the previous word.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
ReturnValues		
Return Value	Type	Meaning
*p	pointer to char	Pointer to new cursor

Table 2.5-36: ScanBackwards CSU [8.5.1.4.29]

2.5.1.5 par_fields.c CSC

/simnet/libsrc/libparser/par_fields.c

This CSC contains the CSUs that maintain the data fields.

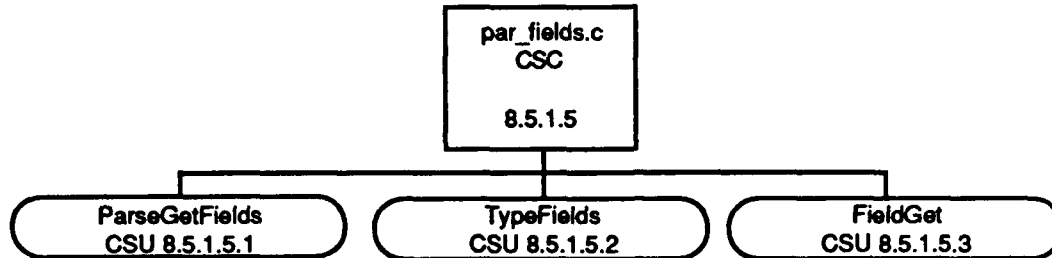


Figure 2.5-6: par_fields.c CSC Structure

2.5.1.5.1 ParseGetFields CSU

This CSU obtains the flag bits.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
argc	int	Standard
argv[]	pointer to char	Standard
ftable	pointer to register PARSE_TABLE	Sec. 2.5.1.11
ReturnValues		
Return Value	Type	Meaning
(argc+2)	int	
PARSE IGNORE	int	
PARSE ERROR	int	
PARSE UPDATE LINE	int	
(argc)	int	
Calls		
Function	Where Described	
ParseGetToken	Sec. 2.5.1.7.6	
ParseError	Sec. 2.5.1.7.7	
ParseTableFind	Sec. 2.5.1.7.11	
ParseEscapeComplete	Sec. 2.5.1.7.12	
FieldGet	Sec. 2.5.1.5.3	
p_arg0	Sec. 2.5.1.11 See Appendix A	
ParseMatch	Sec. 2.5.1.7.10	
ParsePrint	Sec. 2.5.1.9.3	

Table 2.5-37: ParseGetFields CSU [8.5.1.5.1]

2.5.1.5.2 TypeFields CSU

This CSU is used to type the fields symbolically.

Parameters		
Parameters	Type	Where Typedef Declared
ftable	pointer to PARSE_TABLE	Sec. 2.5.1.11
flag	int	Standard
Calls		
Function	Where Described	
ParsePrint	Sec. 2.5.1.9.3	
FieldGet	Sec. 2.5.1.5.3	

Table 2.5-38: TypeFields CSU [8.5.1.5.2]

2.5.1.5.3 FieldGet CSU

This CSU returns the field name associated with a bit pattern.

Parameters		
Parameters	Type	Where Typedef Declared
ftable	pointer to PARSE_TABLE	Sec. 2.5.1.11
bit	int	Standard
ReturnValues		
Return Value	Type	Meaning
(p_arg0(ptr))	pointer to char	Field name
("<<bad field>>")	pointer to char	Field is bad
Calls		
Function	Where Described	
p_arg0	Sec. 2.5.1.11 See Appendix A	

Table 2.5-39: FieldGet CSU [8.5.1.5.3]

2.5.1.6 par_lib.c CSC

/simnet/libsrc/libparser/par_lib.c

This CSC consists of a library of parsing routines. In addition, there are two define tables g_ParseYesNo and g_parseOnOff used by the "yes/no" and "on/off" macros.

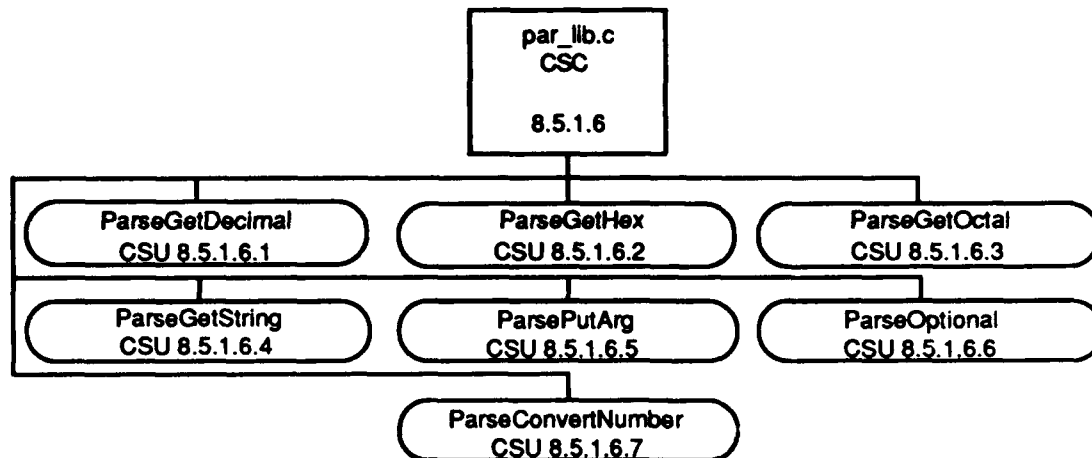


Figure 2.5-7: par_lib.c CSC Structure

2.5.1.6.1 ParseGetDecimal CSU

This CSU returns a decimal number by means of a call to ParseConvertNumber.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
argc	int	Standard
argv[1]	pointer to char	Standard
arg2	pointer to char	Standard
ReturnValues		
Return Value	Type	Meaning
(ParseConvertNumber (argc, argv, pdp, arg2, 10, "decimal"))	int	A parse return status or new argc value
Calls		
Function	Where Described	
ParseConvertNumber	Sec. 2.5.1.6.7	

Table 2.5-40: ParseGetDecimal CSU [8.5.1.6.1]

2.5.1.6.2 ParseGetHex CSU

This CSU returns a hexadecimal number by means of a call to ParseConvertNumber.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
argc	int	Standard
argv[]	pointer to char	Standard
arg2	pointer to char	Standard
ReturnValues		
Return Value	Type	Meaning
(ParseConvertNumber (argc, argv, pdp, arg2, 16, "hex"))	int	A parse return status or new argc value
Calls		
Function	Where Described	
ParseConvertNumber	Sec. 2.5.1.6.7	

Table 2.5-41: ParseGetHex CSU [8.5.1.6.2]

2.5.1.6.3 ParseGetOctal CSU

This CSU returns a octal number by means of a call to ParseConvertNumber.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
argc	int	Standard
argv[]	pointer to char	Standard
arg2	pointer to char	Standard
ReturnValues		
Return Value	Type	Meaning
(ParseConvertNumber (argc, argv, pdp, arg2, 8, "octal"))	int	A parse return status or new argc value
Calls		
Function	Where Described	
ParseConvertNumber	Sec. 2.5.1.6.7	

Table 2.5-42: ParseGetOctal CSU [8.5.1.6.3]

2.5.1.6.4 ParseGetString CSU

This CSU attempts to get a string. The returned values are a function of the value obtained from a call to the CSU ParseGetToken, shown in the "Meaning" column.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
argc	int	Standard
argv[]	pointer to char	Standard
arg2	pointer to char	Standard
ReturnValues		
Return Value	Type	Meaning
argc+1	int	TOKEN_DATA, new value
PARSE_IGNORE	int	TOKEN_ESCAPE or TOKEN_QUESTION
PARSE_ADDSPACE	int	TOKEN_EXHAUSTED
argc	int	TOKEN_OPTIONAL_DONE
ERROR	int	Other value, an internal error
Errors		
Error Name	Reason for Error	
ERROR	Returned due to token return value from ParseGetToken call	
Calls		
Function	Where Described	
ParseGetToken	Sec. 2.5.1.7.6	
stringcopy	Sec. 2.5.1.3.1	
ParseMustFree	Sec. 2.5.1.7.4	
ParseError	Sec. 2.5.1.7.7	
ParsePrint	Sec. 2.5.1.9.3	

Table 2.5-43: ParseGetString CSU [8.5.1.6.4]

2.5.1.6.5 ParsePutArg CSU

This CSU puts an argument into the input stream.

Parameters		
Parameters	Type	Where Typedef Declared
nu_pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
argc	register int	Standard
argv[]	pointer to char	Standard
arg2	pointer to char	Standard
ReturnValues		
Return Value	Type	Meaning
argc+1	int	Status or new argc value

Table 2.5-44: ParsePutArg CSU [8.5.1.6.5]

2.5.1.6.6 ParseOptional CSU

This CSU processes an optional argument by setting the pdp optional element to TRUE and returning the passed argument, argc.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
argc	register int	Standard
nu_argv[]	pointer to char	Standard
nu_arg2	pointer to char	Standard
ReturnValues		
Return Value	Type	Meaning
argc	int	Status or new argc value

Table 2.5-45: ParseOptional CSU [8.5.1.6.6]

2.5.1.6.7 ParseConvertNumber CSU

This CSU obtains a number of a specified base.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
argc	int	Standard
argv[]	pointer to char	Standard
arg2	pointer to char	Standard
base	int	Standard
bname	pointer to char	Standard
ReturnValues		
Return Value	Type	Meaning
PARSE_ERROR	int	Illegal number
argc+1	int	New value
PARSE_ADDSPACE	int	Not a number
PARSE_IGNORE	int	Not a number as expected
argc	int	Old value, option completed
Calls		
Function	Where Described	
ParseGetToken	Sec. 2.5.1.7.6	
ParseError	Sec. 2.5.1.7.7	
ParsePrint	Sec. 2.5.1.9.3	

Table 2.5-46: ParseConvertNumber CSU [8.5.1.6.7]

2.5.1.7 par_util.c CSC

/simnet/libsrc/libparser/par_util.c

This CSC contains parser auxilliary CSUs.

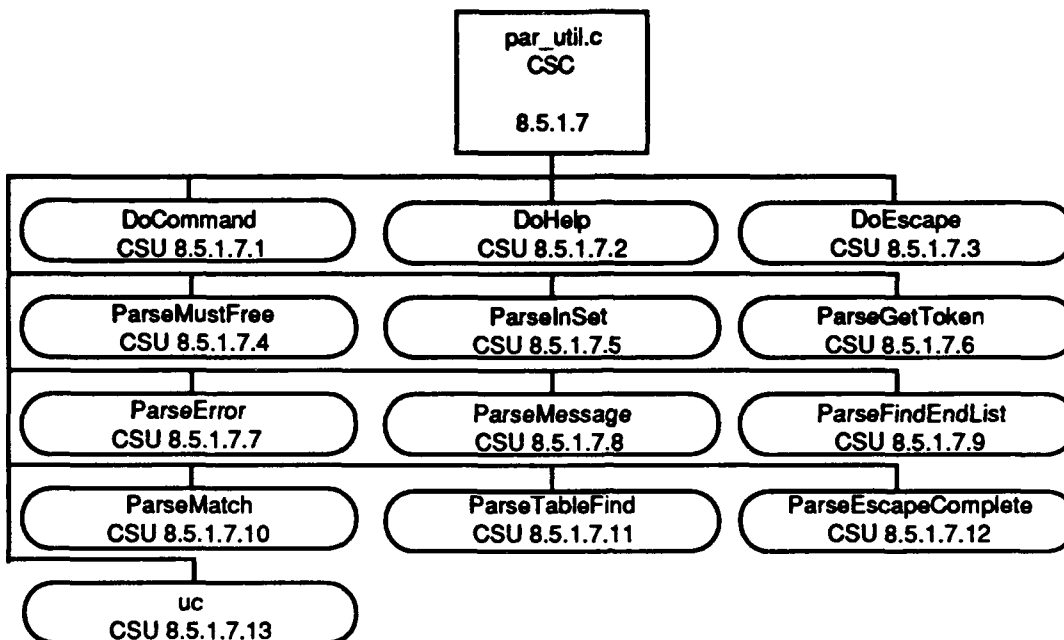


Figure 2.5-8: par_util.c CSC Structure

2.5.1.7.1 DoCommand CSU

This CSU processes a user command.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
Calls		
Function	Where Described	
ParsePrint	Sec. 2.5.1.9.3	
SetStopPoint	Sec. 2.5.1.4.6	
ParseInput	Sec. 2.5.1.1.1	

Table 2.5-47: DoCommand CSU [8.5.1.7.1]

2.5.1.7.2 DoHelp CSU

This CSU processes a help command.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
Calls		
Function	Where Described	
InsertChar	Sec. 2.5.1.4.23	
UpdateLine	Sec. 2.5.1.4.3	
Delete1Backward	Sec. 2.5.1.4.17	
ParsePrint	Sec. 2.5.1.9.3	
SetStopPoint	Sec. 2.5.1.4.6	
ParseInput	Sec. 2.5.1.1.1	

Table 2.5-48: DoHelp CSU [8.5.1.7.2]

2.5.1.7.3 DoEscape CSU

This CSU processes an escape command.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
Calls		
Function	Where Described	
SetStopPoint	Sec. 2.5.1.4.6	
ParseInput	Sec. 2.5.1.1.1	

Table 2.5-49: DoEscape CSU [8.5.1.7.3]

2.5.1.7.4 ParseMustFree CSU

This CSU informs the parse system that the current data item must be freed when the parse is done.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
index	int	Standard

Table 2.5-50: ParseMustFree CSU [8.5.1.7.4]

2.5.1.7.5 ParseInSet CSU

This CSU determines if the character passed is in the set used.

Parameters		
Parameters	Type	Where Typedef Declared
ch	register char	Standard
set	pointer to register char	Standard
ReturnValues		
Return Value	Type	Meaning
TRUE	int	Character is in the set passed
FALSE	int	Character not in set passed

Table 2.5-51: ParseInSet CSU [8.5.1.7.5]

2.5.1.7.6 ParseGetToken CSU

This CSU obtains the next user token and acts upon it.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
tok	pointer to pointer to register char	Standard
ReturnValues		
Return Value	Type	Meaning
TOKEN_OPTIONAL_ EXHAUSTED	int	Token option exhausted
TOKEN EXHAUSTED	int	Token exhausted
pdp->TokenType	int	Token value
TOKEN TYPE	int	Token type
Calls		
Function	Where Described	
MarkCursor	Sec. 2.5.1.4.8	
NextChar	Sec. 2.5.1.4.7	
ParseInSet	Sec. 2.5.1.7.5	

Table 2.5-52: ParseGetToken CSU [8.5.1.7.6]

2.5.1.7.7 ParseError CSU

This CSU processes an error message.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
arg2	pointer to char	Standard
msg1	pointer to char	Standard
msg2	pointer to char	Standard
msg3	pointer to char	Standard
msg4	pointer to char	Standard
msg5	pointer to char	Standard
Calls		
Function	Where Described	
ParsePrint	Sec. 2.5.1.9.3	

Table 2.5-53: ParseError CSU [8.5.1.7.7]

2.5.1.7.8 ParseMessage CSU

This CSU prints a user message.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
message	pointer to char	Standard
Calls		
Function	Where Described	
ParsePrint	Sec. 2.5.1.9.3	

Table 2.5-54: ParseMessage CSU [8.5.1.7.8]

2.5.1.7.9 ParseFindEndList CSU

This CSU finds the end of a parse table block.

Parameters		
Parameters	Type	Where Typedef Declared
first	pointer to register PARSE_TABLE	Sec. 2.5.1.11
end	pointer to register PARSE_TABLE	Sec. 2.5.1.11
begin tok	int	Standard
end tok	int	Standard

ReturnValues		
Return Value	Type	Meaning
first	pointer to PARSE TABLE	Parse table entry point
(PARSE TABLE *) NULL	pointer to PARSE TABLE	End of list not found
Calls		
Function	Where Described	
ParsePrint	Sec. 2.5.1.9.3	

Table 2.5-55: ParseFindEndList CSU [8.5.1.7.9]

2.5.1.7.10 ParseMatch CSU

This CSU determines if a token matches a command string.

Parameters		
Parameters	Type	Where Typedef Declared
command	pointer to register char	Standard
token	pointer to register char	Standard
ReturnValues		
Return Value	Type	Meaning
TRUE	int	Token matches command
FALSE	int	Token not equal command
Calls		
Function	Where Described	
uc	Sec. 2.5.1.7.13	

Table 2.5-56: ParseMatch CSU [8.5.1.7.10]

2.5.1.7.11 ParseTableFind CSU

This CSU searches for a token in a table.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE DATA	Sec. 2.5.1.11
table	pointer to PARSE TABLE	Sec. 2.5.1.11
token	pointer to char	Standard
escape_used	int	Standard
entry	pointer to pointer to PARSE TABLE	Sec. 2.5.1.11

ReturnValues		
Return Value	Type	Meaning
PARSE IGNORE	int	Bad name
PARSE ERROR	int	Bad name
PARSE SUCCESS	int	Successful
Calls		
Function	Where Described	
p_arg0	Sec. 2.5.1.11 See Appendix A	
ParseMatch	Sec. 2.5.1.7.10	
ParseError	Sec. 2.5.1.7.7	

Table 2.5-57: ParseTableFind CSU [8.5.1.7.11]

2.5.1.7.12 ParseEscapeComplete CSU

This CSU updates a token in the input string upon escape.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
token	pointer to register char	Standard
addspace	int	Standard
Calls		
Function	Where Described	
RestoreCursor	Sec. 2.5.1.4.9	
NextChar	Sec. 2.5.1.4.7	
InsertChar	Sec. 2.5.1.4.23	

Table 2.5-58: ParseEscapeComplete CSU [8.5.1.7.12]

2.5.1.7.13 uc CSU

This CSU returns an uppercase character if a lower case one is passed; otherwise, it returns the passed character.

Parameters		
Parameters	Type	Where Typedef Declared
ch	register char	Standard
ReturnValues		
Return Value	Type	Meaning
ch	char	Upper case character

Table 2.5-59: uc CSU [8.5.1.7.13]

2.5.1.8 par_hist.c CSC

/simnet/libsrc/libparser/par_hist.c

This CSC contains CSUs that implement the logging of commands issued by the user to the parser.

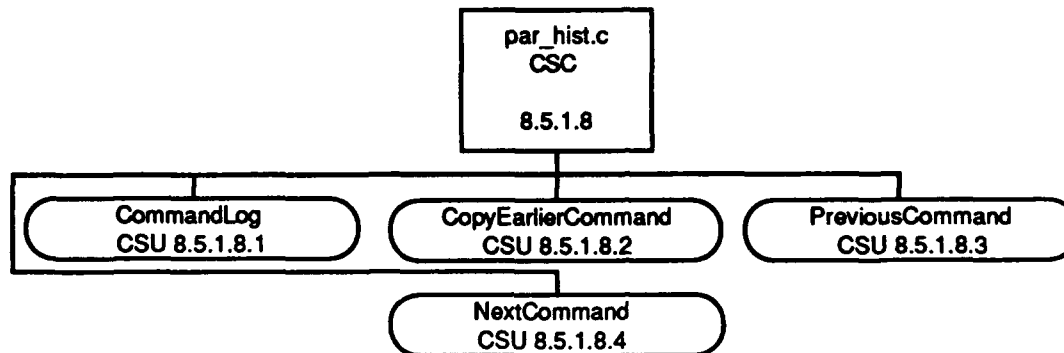


Figure 2.5-9: par_hist.c CSC Structure

2.5.1.8.1 CommandLog CSU

This CSU issues the call to the function that the user invoked from the command line. It copies the command line into a buffer that has a QELEM before the null terminated command line. Then it attaches the buffer to the end of the parse queue.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
Calls		
Function	Where Described	
stringcopy	Sec. 2.5.1.3.1	
Free	Sec. 2.5.1.9.2	

Table 2.5-60: CommandLog CSU [8.5.1.8.1]

2.5.1.8.2 CopyEarlierCommand CSU

This CSU is used to restore the command line pointers and buffer area to those of the "CurrentCommand" QELEM of the global structure.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11

Calls	
Function	Where Described
BeginningOfLine	Sec. 2.5.1.4.11
KillForward	Sec. 2.5.1.4.18
InsertChar	Sec. 2.5.1.4.23

Table 2.5-61: CopyEarlierCommand CSU [8.5.1.8.2]

2.5.1.8.3 PreviousCommand CSU

This CSU travels through the queue using the tail of the queue and the previous QELEM.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
Calls		
Function	Where Described	
CopyEarlierCommand	Sec. 2.5.1.8.2	

Table 2.5-62: PreviousCommand CSU [8.5.1.8.3]

2.5.1.8.4 NextCommand CSU

This CSU travels through the queue using the head of the queue and the next QELEM. The routine checks if the earlier command is the same as the current command, and if so, returns. Otherwise, it increments the earlier command number. If it is not at the end of the command buffer the earlier command is set to the buffer command. If the pointer to the earlier command is NULL, the earlier command is set to the current command. The CSU issues a call to CopyEarlierCommand and returns.

Parameters		
Parameters	Type	Where Typedef Declared
pdp	pointer to register PARSE_DATA	Sec. 2.5.1.11
Calls		
Function	Where Described	
CopyEarlierCommand	Sec. 2.5.1.8.2	

Table 2.5-63: NextCommand CSU [8.5.1.8.4]

2.5.1.9 par_unix.c CSC

/simnet/libsrc/libparser/par_unix.c

This CSC contains two UNIX-based functions, Alloc and Free, along with the ParsePrint function. The contents of the last CSU is conditional on whether ORIGINAL is defined.

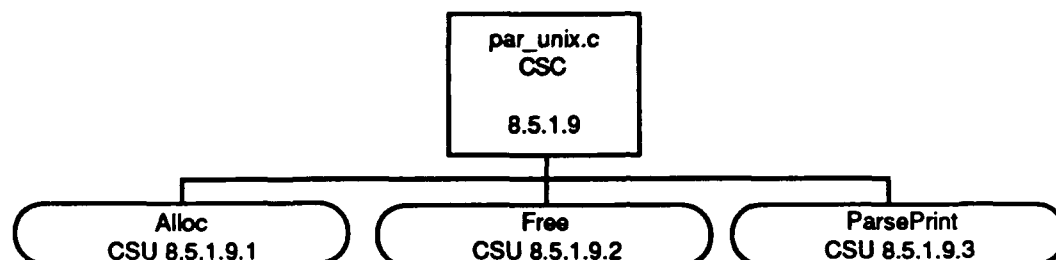


Figure 2.5-10: par_unix.c CSC Structure

2.5.1.9.1 Alloc CSU

This CSU returns the results of a call to heap_calloc, which passes as parameters 1 (number of memory blocks) and size (of that block). The size is the integer parameter passed to Alloc.

Parameters		
Parameters	Type	Where Typedef Declared
size	int	Standard
ReturnValues		
Return Value	Type	Meaning
heap_calloc(1, size)	pointer to char	Pointer to heap
Calls		
Function	Where Described	
heap_calloc	Sec. 2.14.2.1.1	

Table 2.5-64: Alloc CSU [8.5.1.9.1]

2.5.1.9.2 Free CSU

This CSU calls the standard library function free, passing the character pointer.

Parameters		
Parameters	Type	Where Typedef Declared
pointer	pointer to char	Standard
Calls		
Function	Where Described	
free	Sec. 2.5.1.9.2	

Table 2.5-65: Free CSU [8.5.1.9.2]

2.5.1.9.3 ParsePrint CSU

A call to `_doprint` executes the "heart" of the parse printing requirement. If `ORIGINAL` is not defined, the CSU requires that `varargs.h` be included. Only if the `ORIGINAL` define is missing are the functions `va_start` and `va_end` called. In addition, the second parameter passed becomes `va_alist`, defined `va_dcl`, instead of the `args` array.

Parameters		
Parameters	Type	Where Typedef Declared
<code>ctl</code>	pointer to char	Standard
<code>args[]</code>	int	Standard

Table 2.5-66: ParsePrint CSU [8.5.1.9.3]

2.5.1.10 kludge.c CSU

`/simnet/libsrc/libparser/kludge.c`

This file consists of macros expanding calls to `printf` statements that effectively print the names of the character strings. These consist of: `ecmderr`, `gethost`, `isbadhost`, `hostname`, `getnet`, `isbadnet`, and `netfmt`. They are expanded in Appendix A.

2.5.1.11 libparser.h CSU

`/simnet/libsrc/libparser/libparser.h`

This CSU is the parser header file. It contains constant definitions, various tokens and function return values, definitions to create parse tables (see Appendix A), and two structure definitions, `PARSE_TABLE` and `PARSE_DATA`. The following tables contain the constant definitions and the two structure definitions.

The parser constant definitions are used in the `PARSE_DATA` structure to define array sizes.

Constant	Value
<code>ARGSIZE</code>	30
<code>INPUTLINE</code>	140
<code>STRINGSIZE</code>	200

Table 2.5-67: Parser Constant Definitions

The following definitions are the legal parse function return values.

Constant	Value
PARSE_ERROR	-1 /* Error with cursor positioning */
PARSE_SUCCESS	-2 /* Cleanup and start new command */
PARSE_IGNORE	-3 /* Error without cursor reposition */
PARSE_NEEDKEYWORD	-4 /* Waiting for another keyword */
PARSE_UPDATE_LINE	-5 /* Just update line and try again */
PARSE_ADDSPACE	-6 /* Add space to end of line */

Table 2.5-68: Parser Function Return Constant Definitions

The following are the legal ParseGetToken, Sec. 2.5.1.7.6, function returns.

Constant	Value
TOKEN_DATA	1 /* A token is returned */
TOKEN_ESCAPE	2 /* A token is returned, it ended in escape */
TOKEN_QUESTION	3 /* A token is returned, it ended in '?' */
TOKEN_EXHAUSTED	4 /* No token is returned, input exhausted */
TOKEN_OPTION_DONE	5 /* Optional input exhausted */

Table 2.5-69: ParseGetToken Return Constant Definitions

Constant	Value
FIELD_ADD	1 /* Add to current value */
FIELD_SUBTRACT	2 /* Subtract from current value */
FIELD_REPLACE	3 /* Replace with current value */

Table 2.5-70: FIELD Change Constant Definitions

Constant	Value
P_CALL	1
P_END_KEYWORD	2
P_END_KEYWORD_SELECT	3
P_KEYWORD	4
P_KEYWORD_SELECT	5
P_PARSE_FUNCTION	6
P_PARSE_KEYWORDS	7
P_PRINT	8
P_PRINTFIELD	9

Table 2.5-71: Command Token Constant Definitions

Item	Type	Where Type Defined
p_command	int	Standard
p_arg1	pointer to char	Standard
p_arg2	pointer to char	Standard

Table 2.5-72: PARSE_TABLE Structure Definition

Item	Type	Where Type Defined
TopLevel	pointer to PARSE_TABLE	Previous structure
TokenPtr	pointer to char	Standard
TokenType	int	Standard
Optional	int	Standard
CommandBuffer	pointer to pointer to char	Standard
CommandBufferEnd	pointer to pointer to char	Standard
EarlierCommand	pointer to pointer to char	Standard
CurrentCommand	pointer to pointer to char	Standard
First	pointer to char	Standard
Cur	pointer to char	Standard
Last	pointer to char	Standard
End	pointer to char	Standard
SaveCur	pointer to char	Standard
CursorMark	pointer to char	Standard
Prompt	pointer to char	Standard
StopPoint	pointer to char	Standard
PrintArg	long	Standard
LogFunction	FUNC_PTR	
Argv[ARGSIZE]	char	Standard
ArgFree[ARGSIZE]	char	Standard
ParseToken[INPUTLINE]	char	Standard
InputBuffer[INPUTLINE]	char	Standard
SaveBuffer[INPUTLINE]	char	Standard
StringBuffer[STRINGSIZE]	char	Standard

Table 2.5-73: PARSE_DATA Structure Definition

2.5.2 Parser Command Processor CSC

2.5.2.1 parser.c CSC

/simnet/src/host/parser.c

This CSC contains the code which handles user input, sending it directly to the program from the terminal window where the phantom was started. It is used mainly to perform tasks that the ordinary user would not want to perform, or should not be allowed to perform. Almost all system information is available through the parser, as well as many debugging and development aids.



Figure 2.5-11: parser.c CSC Structure

This CSC begins by undefining `ENABLE_ADB`, important because of conditional compiles in some of the CSUs. A constant definition then follows.

Constant	Value
PARSER_STRING_SIZE	128

Table 2-74: PARSER_STRING_SIZE Constant Definition

This constant definition defines the size of the `parser_title[]` and `parser_hostname[]` arrays. Then locally used variables are initialized, `stealth_addr` (type `SimulationAddress`) is set to zero {0, 0}, the unsigned int `g_stealth_draw_tick` is set to 5000, and `g_stealth_mimic` is set to TRUE.

2.5.2.1.1 parser_init CSU

This CSU strips off the parser host name and prints it in upper case letters.

Calls	
Function	Where Described
<code>gethostname</code>	Sec. 2.5.1.10
<code>upcase</code>	Sec. 2.14.3.8.1
<code>tty_parser_init</code>	<code>libtty</code>

Table 2.5-75: parser_init CSU [8.5.2.1.1]

2.5.2.1.2 parser_restore_term CSU

If `ENABLE_ADB` is not defined, this CSU restores initial terminal conditions through a call to `tty_exit()`.

Calls	
Function	Where Described
<code>tty_exit</code>	<code>libtty</code>

Table 2.5-76: parser_restore_term CSU [8.5.2.1.2]

2.5.2.1.3 parser_create CSU

If `ENABLE_ADB` is not defined, this CSU creates a parser through calls to `parser_init()` and `periodic_fncl()`.

Parameters		
Parameters	Type	Where Typedef Declared
<code>title</code>	pointer to char	Standard
<code>prompt</code>	pointer to char	Standard
<code>make_current</code>	int	Standard

Calls	
Function	Where Described
parser_init	Sec. 2.5.2.1.1
periodic_fnc1	Sec. 2.2.1.1.2
tty_tick	libtty

Table 2.5-77: parser_create CSU [8.5.2.1.3]

2.5.2.1.4 set_command_printing CSU

This CSU sets g_print_command to TRUE (to print) if argv[0] is TRUE (ON); otherwise, it sets g_print_command to FALSE.

Parameters		
Parameters	Type	Where Typedef Declared
argc	int	Standard
argv[]	int	Standard

Table 2.5-78: set_command_printing CSU [8.5.2.1.4]

2.5.2.1.5 set_abort_on_error CSU

This CSU sets the variable g_abort_on_error to the value passed in argv[0].

Parameters		
Parameters	Type	Where Typedef Declared
argc	int	Standard
argv[]	int	Standard

Table 2.5-79: set_abort_on_error CSU [8.5.2.1.5]

2.5.2.1.6 set_ground_impact_mode CSU

This CSU sets g_sbx_show_gi to TRUE if argv[0] is TRUE (ON); otherwise, it sets g_sbx_show_gi to FALSE. The CSU also prints that it is turning the ground impact mode (gi_mode) on or off as appropriate.

Parameters		
Parameters	Type	Where Typedef Declared
argc	int	Standard
argv[]	int	Standard

Table 2.5-80: set_ground_impact_mode CSU [8.5.2.1.6]

2.5.2.1.7 set_indirect_fire_mode CSU

This CSU sets `g_sbx_show_if` to TRUE if `argv[0]` is TRUE (ON); otherwise, it sets `g_sbx_show_if` to FALSE. The CSU also prints that it is turning the indirect fire mode (`ind_fire_mode`) on or off as appropriate.

Parameters		
Parameters	Type	Where Typedef Declared
<code>argc</code>	int	Standard
<code>argv[]</code>	int	Standard

Table 2.5-81: set_indirect_fire_mode CSU [8.5.2.1.7]

2.5.2.1.8 set_header_printing CSU

This CSU sets `g_print_headers` to TRUE if `argv[0]` is TRUE (ON); otherwise, it sets `g_print_headers` to FALSE.

Parameters		
Parameters	Type	Where Typedef Declared
<code>argc</code>	int	Standard
<code>argv[]</code>	int	Standard

Table 2.5-82: set_header_printing CSU [8.5.2.1.8]

2.5.2.1.9 set_monitor_period CSU

This CSU turns on the performance monitor if `argv[0]` contains a non-zero value. The monitoring frequency is contained in `argv[0]` in seconds. The period, in milliseconds, is passed in a call to `perf_monitor_on()`. If `argv[0]` is zero, the performance monitor is turned off via a call to `perf_monitor_off()`. In either case, an appropriate message is displayed to the user.

Parameters		
Parameters	Type	Where Typedef Declared
<code>argc</code>	int	Standard
<code>argv[]</code>	int	Standard
Calls		
Function	Where Described	
<code>perf_monitor_on</code>	Sec. 2.2.1.4.1	
<code>perf_monitor_off</code>	Sec. 2.2.1.4.2	

Table 2.5-83: set_monitor_period CSU [8.5.2.1.9]

2.5.2.1.10 toggle_debugging CSU

This CSU toggles the `g_debug_flags` variable as a function of the current conditions of `argv[1]` and `argv[2]`.

Parameters		
Parameters	Type	Where Typedef Declared
<code>argc</code>	int	Standard
<code>argv[]</code>	int	Standard

Table 2.5-84: toggle_debugging CSU [8.5.2.1.10]

2.5.2.1.11 show_vehicle_ids CSU

This CSU prints vehicle information on the composites, local vehicles, stealth local vehicles, local missiles, and remote vehicles (including those that timed out or deactivated).

Parameters		
Parameters	Type	Where Typedef Declared
<code>argc</code>	int	Standard
<code>argv[]</code>	int	Standard
Calls		
Function	Where Described	
<code>vehicle_manager print</code>	Sec. 2.9.1.3.4	

Table 2.5-85: show_vehicle_ids CSU [8.5.2.1.11]

2.5.2.1.12 count_vehicles CSU

This CSU counts and prints the number of vehicles, including composites, local vehicles, stealth local vehicles, local missiles, and remote vehicles (including those that timed out or deactivated).

Parameters		
Parameters	Type	Where Typedef Declared
<code>argc</code>	int	Standard
<code>argv[]</code>	int	Standard
Calls		
Function	Where Described	
<code>vehicle_manager count</code>	Sec. 2.9.3.1.5	

Table 2.5-86: count_vehicles CSU [8.5.2.1.12]

2.5.2.1.13 count_forces CSU

This CSU counts and prints the number of forces in each category, including distinguished, other, observer, and target.

Parameters		
Parameters	Type	Where Typedef Declared
argc	int	Standard
argv[]	int	Standard
Calls		
Function	Where Described	
vehicle_manager_count_forces	Sec. 2.9.3.1.6	

Table 2.5-87: count_forces CSU [8.5.2.1.13]

Prior to the count_sites CSU, the parcer.c CSC contains the following constant defines for the number of sites and hosts.

Constant	Value
NUMBER OF SITES	65536
NUMBER OF HOSTS	65536

Table 2.5-88: Sites and Hosts Constant Definitions

2.5.2.1.14 count_sites CSU

This CSU counts and prints the number of vehicles at each of the sites.

Parameters		
Parameters	Type	Where Typedef Declared
argc	int	Standard
argv[]	int	Standard
Calls		
Function	Where Described	
vehicle_manager_count_sites	Sec. 2.9.3.1.7	

Table 2.5-89: count_sites CSU [8.5.2.1.14]

2.5.2.1.15 count_hosts CSU

This CSU counts and prints the number of vehicles on each of the hosts at a site.

Parameters		
Parameters	Type	Where Typedef Declared
argc	int	Standard
argv[]	int	Standard

Calls	
Function	Where Described
vehicle_manager_count_hosts	Sec. 2.9.3.1.8

Table 2.5-90: count_hosts CSU [8.5.2.1.15]

2.5.2.1.16 show_connection CSU

This CSU calls show_connection_all_sbx, which causes all open ports to display status information.

Parameters		
Parameters	Type	Where Typedef Declared
argc	int	Standard
argv[]	int	Standard
Calls		
Function	Where Described	
show connection all sbx	Sec. 2.4.3.2.4	

Table 2.5-91: show_connection CSU [8.5.2.1.16]

2.5.2.1.17 show_sbx_overlays CSU

This CSU calls show_overlays_all_sbx, which causes all overlays associated with each port to be displayed.

Parameters		
Parameters	Type	Where Typedef Declared
argc	int	Standard
argv[]	int	Standard
Calls		
Function	Where Described	
show overlays all sbx	Sec. 2.4.3.2.7	

Table 2.5-92: show_sbx_overlays CSU [8.5.2.1.17]

2.5.2.1.18 identify_exercise CSU

This CSU prints the exercise identification, including whether or not it is using the SIMNET LAN.

Parameters		
Parameters	Type	Where Typedef Declared
argc	int	Standard
argv[]	int	Standard

Table 2.5-93: identify_exercise CSU [8.5.2.1.18]

2.5.2.1.19 lookup_vehicle_with_range_check CSU

This CSU checks if the identification number of a vehicle is within the vehicle range. If the vehicle id is within range, it checks both type and id, and returns the results of the check. If the vehicle id is too high, it returns NULL.

Parameters		
Parameters	Type	Where Typedef Declared
id	unsigned int	Standard
type	int	Standard
ReturnValues		
Return Value	Type	Meaning
LOOKUP_VEHICLE(id, type)	pointer to SAF OBJECT	Vehicle status
NULL	pointer to SAF OBJECT	Vehicle id too high
Calls		
Function	Where Described	
LOOKUP_VEHICLE	Sec. 2.9.3.2 See Appendix A	

Table 2.5-94: lookup_vehicle_with_range_check CSU [8.5.2.1.19]

2.5.2.1.20 show_vehicle CSU

This CSU determines if a vehicle passed in argv[0] exists. If it does not exist, the CSU prints that it can not locate it. If it does exist, it calls show.

Parameters		
Parameters	Type	Where Typedef Declared
argc	int	Standard
argv[]	int	Standard
Calls		
Function	Where Described	
lookup_vehicle_with_range_check	Sec. 2.5.2.1.19	
show	Sec. 2.14.1.1.4	

Table 2.5-95: show_vehicle CSU [8.5.2.1.20]

2.5.2.1.21 vehicle_catastrophe CSU

This CSU calls `vehicle_kill`, passing `argv[0]` as the vehicle to be killed.

Parameters		
Parameters	Type	Where Typedef Declared
<code>argc</code>	int	Standard
<code>argv[]</code>	int	Standard
Calls		
Function	Where Described	
<code>vehicle_kill</code>	Sec. 2.3.1.2	

Table 2.5-96: vehicle_catastrophe CSU [8.5.2.1.21]

2.5.2.1.22 vehicle_resupply CSU

This CSU determines if a local vehicle passed in `argv[0]` exists. If it does not exist, the CSU prints that it can not locate it. If it does exist, it calls `start_resupply_of_to`.

Parameters		
Parameters	Type	Where Typedef Declared
<code>argc</code>	int	Standard
<code>argv[]</code>	int	Standard
Calls		
Function	Where Described	
<code>lookup_vehicle_with_range_check</code>	Sec. 2.5.2.1.19	
<code>start_resupply_of_to</code>	Sec. 2.6.6.1.4	

Table 2.5-97: vehicle_resupply CSU [8.5.2.1.22]

2.5.2.1.23 vehicle_fake_resupply CSU

This CSU determines if a local unit passed in `argv[0]` exists. If it does not exist, the CSU prints that it can not locate it. If it does exist, it calls `fake_resupply`.

Parameters		
Parameters	Type	Where Typedef Declared
<code>argc</code>	int	Standard
<code>argv[]</code>	int	Standard
Calls		
Function	Where Described	
<code>lookup_vehicle_with_range_check</code>	Sec. 2.5.2.1.19	
<code>fake_resupply</code>	Sec. 2.14.1.1.12	

Table 2.5-98: vehicle_fake_resupply CSU [8.5.2.1.23]

2.5.2.1.24 vehicle_defuel CSU

This CSU determines if a local vehicle passed in argv[0] exists. If it does not exist, the CSU prints that it can not locate it. If it does exist, it sets the range remaining (vehicle -> saf_vehicle -> range_remaining) to zero.

Parameters		
Parameters	Type	Where Typedef Declared
argc	int	Standard
argv[]	int	Standard
Calls		
Function	Where Described	
lookup_vehicle_with_range_check	Sec. 2.5.2.1.19	

Table 2.5-99: vehicle_defuel CSU [8.5.2.1.24]

2.5.2.1.25 vehicle_ping_do CSU

This CSU calls vehicle_ping, which applies a mild vehicle impact.

Parameters		
Parameters	Type	Where Typedef Declared
argc	int	Standard
argv[]	int	Standard
Calls		
Function	Where Described	
vehicle_ping	Sec. 2.3.1.6	

Table 2.5-100: vehicle_ping_do CSU [8.5.2.1.25]

2.5.2.1.26 vehicle_bong_do CSU

This CSU calls vehicle_bong, which applies a deadly vehicle impact.

Parameters		
Parameters	Type	Where Typedef Declared
argc	int	Standard
argv[]	int	Standard
Calls		
Function	Where Described	
vehicle_bong	Sec. 2.3.1.5	

Table 2.5-101: vehicle_bong_do CSU [8.5.2.1.26]

2.5.2.1.27 stealth_set_symbols_draw_tick CSU

This CSU sets the stealth updating time (g_stealth_draw_tick) to the value passed in argv[0], adjusted to millisecond resolution.

Parameters		
Parameters	Type	Where Typedef Declared
argc	int	Standard
argv[]	int	Standard

Table 2.5-102: stealth_set_symbols_draw_tick CSU [8.5.2.1.27]

2.5.2.1.28 stealth_set_mimic_on CSU

This CSU sets g_stealth_mimic to TRUE.

2.5.2.1.29 stealth_set_mimic_off CSU

This CSU sets g_stealth_mimic to FALSE.

2.5.2.1.30 stealth_site_host_pair CSU

This CSU sets the stealth address site and host pair from parameters passed in argv[].

Parameters		
Parameters	Type	Where Typedef Declared
argc	int	Standard
argv[]	int	Standard

Table 2.5-103: stealth_site_host_pair CSU [8.5.2.1.30]

2.5.2.1.31 stealth_attach_to CSU

This CSU determines if a local vehicle passed in argv[0] exists. If it does not exist, the CSU prints that it can not locate it. If it does exist, it tests to see if the stealth site host pair exists. If the host pair does not exist, it requests the stealth address. If the site does exist, the vehicle is "teleported" to the site, avoiding collision, and is attached to the vehicle there.

Parameters		
Parameters	Type	Where Typedef Declared
argc	int	Standard
argv[]	int	Standard

Calls	
Function	Where Described
lookup_vehicle_with_range_check	Sec. 2.5.2.1.19
vec copy	Sec. 2.14.3.5.17
OBJ POSITON	Sec. 2.9.1.1 See Appendix A
stealth_send_teleport_to	
stealth_send_attach_to_vehicle	

Table 2.5-104: stealth_attach_to CSU [8.5.2.1.31]

2.5.2.1.32 stealth_teleport_to CSU

This CSU determines if the stealth site host pair exists. If the host pair does not exist, it requests the stealth address. If the site does exist, the vehicle is "teleported" to the site.

Parameters		
Parameters	Type	Where Typedef Declared
argc	int	Standard
argv[]	int	Standard
Calls		
Function	Where Described	
tdb_get_z	Sec. 2.14.1.2.3	
deg to rad	sim macros.h	
stealth_send_teleport_to		

Table 2.5-105: stealth_teleport_to CSU [8.5.2.1.32]

2.5.2.1.33 parser_create_vehicle CSU

This CSU initializes all the parameters for a vehicle. It creates the vehicle as a top level unit, updating the owner's top level unit information.

Parameters		
Parameters	Type	Where Typedef Declared
argc	int	Standard
argv[]	int	Standard
Calls		
Function	Where Described	
get_symbol	Sec. 2.1.1.3.2	
tdb_get_gl	Sec. 2.141.1.2.2	
countries_from_battle_scheme_and_force	Sec. 2.5.2.1.34	
get_sbx_from_port	Sec. 2.4.3.2.5	
create_unit	Sec. 2.11.1.2	

Table 2.5-106: parser_create_vehicle CSU [8.5.2.1.33]

2.5.2.1.34 countries_from_battle_scheme_and_force CSU

This CSU sets the countries for the offensive and defensive forces according to the battle scheme and force identification passed.

Parameters		
Parameters	Type	Where Typedef Declared
battle_scheme	BattleScheme	
forceID	ForceID	
cD	pointer to unsigned char	Standard
cO	pointer to unsigned char	Standard
Calls		
Function	Where Described	
ERROR_OUT	Sec. 2.5.2.2	

Table 2.5-107: countries from battle_scheme_and_force CSU [8.5.2.1.34]

2.5.2.1.35 parser_global_reset CSU

This CSU calls saf_complete_reset, which completely resets all current SAF vehicles.

Parameters		
Parameters	Type	Where Typedef Declared
argc	int	Standard
argv[]	int	Standard
Calls		
Function	Where Described	
saf complete reset	Sec. 2.2.4.1	

Table 2.5-108: parser_global_reset CSU [8.5.2.1.35]

At this point in the parser.c CSC, two additional constants are defined for use by the remaining CSUs.

Constant	Value
STAT_LONG	0x01
STAT_SHORT	0x10

Table 2.5-109: STAT_LONG and STAT_SHORT Constant Definitions

2.5.2.1.36 parser_heap_statistics CSU

This CSU determines, from the flag passed in argv[1], whether or not to do a long version of the heap statistics. The decision is passed as a parameter in a call to do_heap_statistics, which does the actual work.

Parameters		
Parameters	Type	Where Typedef Declared
argc	int	Standard
argv[]	int	Standard
Calls		
Function	Where Described	
do_heap_statistics	Sec. 2.14.2.1.17	

Table 2.5-110: parser_heap_statistics CSU [8.5.2.1.36]

2.5.2.1.37 parser_heap_verify CSU

This CSU calls do_heap_verify.

Parameters		
Parameters	Type	Where Typedef Declared
argc	int	Standard
argv[]	int	Standard
Calls		
Function	Where Described	
do_heap_verify	Sec. 2.14.2.1.16	

Table 2.5-111: parser_heap_verify CSU [8.5.2.1.37]

2.5.2.1.38 print_reasons_and_clear CSU

This CSU calls print_reasons followed by clear_monitor_variables.

Parameters		
Parameters	Type	Where Typedef Declared
argc	int	Standard
argv[]	int	Standard
Calls		
Function	Where Described	
print_reasons		
clear_monitor_variables		

Table 2.5-112: print_reasons_and_clear CSU [8.5.2.1.38]

2.5.2.1.39 parser_heap_collect CSU

This CSU calls `do_heap_collect`.

Parameters		
Parameters	Type	Where Typedef Declared
<code>argc</code>	int	Standard
<code>argv[]</code>	int	Standard
Calls		
Function	Where Described	
<code>do_heap_collect</code>	Sec. 2.14.2.1.18	

Table 2.5-113: parser_heap_collect CSU [8.5.2.1.39]

2.5.2.1.40 parser_set_targeting_parameters CSU

This CSU determines if the specified unit exists. If it does not exist, it prints a message stating this fact. If the unit does exist, it calls `set_targeting_parameters` to set the targeting parameters.

Parameters		
Parameters	Type	Where Typedef Declared
<code>argc</code>	int	Standard
<code>argv[]</code>	int	Standard
Calls		
Function	Where Described	
<code>lookup_vehicle_with_range_check</code>	Sec. 2.5.2.1.19	
<code>set_targeting_parameters</code>	Sec. 2.14.1.1.7	

Table 2.5-114: parser_set_targeting_parameters CSU [8.5.2.1.40]

2.5.2.1.41 parser_send_string CSU

This CSU sends a port number message passed as a parameter in a call to `sbx_printf`.

Parameters		
Parameters	Type	Where Typedef Declared
<code>argc</code>	int	Standard
<code>argv[]</code>	int	Standard
Calls		
Function	Where Described	
<code>sbx_printf</code>	Sec. 2.4.3.2.8	

Table 2.5-115: parser_send_string CSU [8.5.2.1.41]

The remaining portion of the parser.c CSC consists of executable parser table macros. The expansions for these macros are in libparser.h (Sec. 2.5.1.11), and the macros are shown with their expansions in Appendix A. The constants that compose the debugging (D_) and show command modifiers for these macros are in the tables in debug.h (Sec. 2.5.2.2).

```

FIELD_TABLE (debugging_table)
FIELD ("collisions", D_COLLISION, "- collision detector")
FIELD ("composite", D_COMPOSITE, "- composite unit debugging")
FIELD ("connection", D_CONNECTION, "- sbx connection debugging")
FIELD ("cm", D_CM, "- control measures and overlays")
FIELD ("driver", D_DRIVER, "- driver debugging")
FIELD ("global", D_GLOBAL, "- all debugging")
FIELD ("gunner", D_GUNNER, "- gunner debugging")
FIELD ("intervis", D_INTERVIS, "- intervisibility")
FIELD ("reporter", D_REPORTER, "- report generation")
FIELD ("loader", D_LOADER, "- loader debugging")
FIELD ("missile", D_MISSILE, "- missile debugging")
FIELD ("navigator", D_NAVIGATOR, "- navigator")
FIELD ("obstacles", D_OBSTACLE, "- obstacle avoider")
FIELD ("pilot", D_PILOT, "- pilot debugging")
FIELD ("remote", D_REMOTE, "- remote vehicle debugging")
FIELD ("station", D_STATION, "- station keeper")
FIELD ("tactical", D_TACTICAL, "- tactical state")
FIELD ("targeting", D_TARGETING, "- targeting debugging")
FIELD ("turret", D_TURRET, "- turret debugging")
FIELD ("vehicle", D_VEHICLE, "- vehicle debugging")
FIELD ("weapons", D_WEAPON, "- weapon systems")
FIELD ("groundveh", D_GROUND, "- ground vehicle")
FIELD ("driver", D_DRIVER, "- driver")
FIELD ("airveh", D_AIR, "- air vehicle")
END FIELD_TABLE

```

Table 2.5-116: FIELD_TABLE (debugging_table)

```
DEFINE_TABLE (set_table)
  KEYWORD_SELECT (" Options for the set command")

    KEYWORD ("command_printing", "- printing of SBX commands on/off")
      GET_ON_OR_OFF
      CALL (set_command_printing)
    END_KEYWORD

    KEYWORD ("abort_on_error", "- if on error checks will call abort")
      GET_ON_OR_OFF
      CALL (set_abort_on_error)
    END_KEYWORD

    KEYWORD ("ground_impact_mode", "- display of ground impacts on/off")
      GET_ON_OR_OFF
      CALL (set_ground_impact_mode)
    END_KEYWORD

    KEYWORD ("header_printing", "- printing of SBX headers on/off")
      GET_ON_OR_OFF
      CALL (set_header_printing)
    END_KEYWORD

    KEYWORD ("indirect_fire_mode", "- display of indirect fire on/off")
      GET_ON_OR_OFF
      CALL (set_indirect_fire_mode)
    END_KEYWORD

    KEYWORD ("monitor_period", "- period for scheduler monitor")
      GETDECIMAL("period in seconds")
      CALL (set_monitor_period)
    END_KEYWORD

  END_KEYWORD_SELECT
END_DEFINE_TABLE
```

Table 2.5-117: DEFINE_TABLE (set_table)

```
DEFINE_TABLE (net_table)
  KEYWORD_SELECT (" Options for the net command")

    KEYWORD ("stats", "- print simnet statistics")
      CALL (simnet_getstats)
    END_KEYWORD

    KEYWORD ("zero", "- zero out the simnet statistics")
      CALL (simnet_zerostats)
    END_KEYWORD

    KEYWORD ("thresholding", "- get thresholding statistics")
      CALL (print_reasons_and_clear)
    END_KEYWORD

  END_KEYWORD_SELECT
END_DEFINE_TABLE
```

Table 2.5-118: DEFINE_TABLE (net_table)

```

FIELD_TABLE (heap_stat_table)
  FIELD ("long", STAT_LONG, "- long version of statistics")
  FIELD ("short", STAT_SHORT, "- short versions of statistics")
END FIELD_TABLE

```

Table 2.5-119: DEFINE_TABLE (heap_stat_table)

```

DEFINE_TABLE (hashing_table)
  KEYWORD_SELECT(" Options for the hashing command")

    KEYWORD ("collect", "- start collecting id hashing statistics")
      CALL (start_collecting_id_hashing_statistics)
    END_KEYWORD

    KEYWORD ("report", "- report collected id hashing statistics")
      CALL (report_id_hashing_statistics)
    END_KEYWORD

  END_KEYWORD_SELECT
END DEFINE_TABLE

```

Table 2.5-120: DEFINE_TABLE (hashing_table)

```

DEFINE_TABLE (heap_table)
  KEYWORD_SELECT(" Options for the heap command")

    KEYWORD ("collect", "- force a collect on a heap")
      CALL (parser_heap_collect)
    END_KEYWORD

    KEYWORD ("statistics", "- print statistics for a heap")
      GETFIELDS (heap_stat_table)
      CALL (parser_heap_statistics)
    END_KEYWORD

    KEYWORD ("verify", "- verify the consistency of a heap")
      CALL (parser_heap_verify)
    END_KEYWORD

  END_KEYWORD_SELECT
END DEFINE_TABLE

```

Table 2.5-121: DEFINE_TABLE (heap_table)

```

DEFINE_TABLE (print_table)
  KEYWORD_SELECT(" Options for the print command")

    KEYWORD ("buffers", "- number of buffers allocated")
      CALL (buffer_statistics_print)
    END_KEYWORD

    KEYWORD ("connections", "- connection status")
      CALL (show_connection)
    END_KEYWORD

    KEYWORD ("count", "- count of vehicles in simnet")
      CALL (count_vehicles)
    END_KEYWORD

    KEYWORD ("exercise", "- exercise that this phantom is running in ")
      CALL (identify_exercise)
    END_KEYWORD

    KEYWORD ("hosts", "- count of vehicles by hosts")
      GETDECIMAL("site from which to look at hosts")
      CALL (count_hosts)
    END_KEYWORD

    KEYWORD ("forces", "- count of vehicles by force")
      CALL (count_forces)
    END_KEYWORD

    KEYWORD ("overlays", "- overlays")
      CALL (show_sbx_overlays)
    END_KEYWORD

    KEYWORD ("sites", "- count of vehicles by site")
      CALL (count_sites)
    END_KEYWORD

    KEYWORD ("vehicles", "- id's of current vehicles")
      CALL (show_vehicle_ids)
    END_KEYWORD

    KEYWORD ("version", "- version of phantom program")
      CALL (identify_version)
    END_KEYWORD

  END_KEYWORD_SELECT
END_DEFINE_TABLE

```

Table 2.5-122: DEFINE_TABLE (print_table)

```

DEFINE_TABLE (parser_reset_table)
  KEYWORD_SELECT(" Please confirm...")

    KEYWORD ("yes", "- Reset the phantom (WARNING: THIS CLEARS EVERYTHING)")
      CALL (parser_global_reset)
    END_KEYWORD

    KEYWORD ("no", "- Abort reset operation")
    END_KEYWORD

  END_KEYWORD_SELECT
END_DEFINE_TABLE

```

Table 2.5-123: DEFINE_TABLE (parser_reset_table)


```

FIELD_TABLE (show_table)
  FIELD ("all", S_ALL, "- all available information")
  FIELD ("composite", S_COMPOSITE, "- composite unit status")
  FIELD ("detection", S_DETECTION, "- detection status")
  FIELD ("default", S_DEFAULT, "- basic vehicle status")
  FIELD ("entity", S_ENTITY, "- entity status")
  FIELD ("airveh", S_AIR, "- air vehicle status")
  FIELD ("formation", S_FORMATION, "- formation status")
  FIELD ("gunner", S_GUNNER, "- gunner status")
  FIELD ("intervis", S_INTERVIS, "- visibility status")
  FIELD ("navigator", S_NAVIGATOR, "- navigator status")
  FIELD ("safent", S_SAFENT, "- local entity status")
  FIELD ("safveh", S_SAFVEH, "- local vehicle status")
  FIELD ("spotter", S_SPOTTER, "- spotter status")
  FIELD ("pilot", S_PILOT, "- pilot status")
  FIELD ("remote", S_REMOTE, "- remote vehicle status")
  FIELD ("targeting", S_TARGETING, "- targeting status")
  FIELD ("turret", S_TURRET, "- turret status")
  FIELD ("vehicle", S_VEHICLE, "- vehicle status")
  FIELD ("weapons", S_WEAPON, "- weapons status")
  FIELD ("tactical", S_TACTICAL, "- tactical state")
  FIELD ("report", S_REPORTER, "- report status")
  FIELD ("driver", S_DRIVER, "- driver")
END FIELD_TABLE

```

Table 2.5-124: FIELD_TABLE (show_table)

```

FIELD_TABLE (firestatus_table)
  FIELD ("hold", FIRESTATUS_HOLD_FIRE, "- Hold fire")
  FIELD ("fire", FIRESTATUS_FIRE_AT_WILL, "- Fire at will")
END FIELD_TABLE

```

Table 2.5-125: DEFINE_TABLE (firestatus_table)

```

FIELD_TABLE (create_side_table)
  FIELD ("blue", distinguishedForceID,
    "- US")
  FIELD ("red", otherForceID,
    "- USSR")
END FIELD_TABLE

```

Table 2.5-126: DEFINE_TABLE (create_side_table)

```

DEFINE_TABLE (stealth_mode)
  KEYWORD_SELECT (" Options for stealth mimic mode")

    KEYWORD ("On", "- mimic mode on when stealth attached")
      CALL (stealth_set_mimic_on)
    END_KEYWORD

    KEYWORD ("Off", "- mimic mode off when stealth attached")
      CALL (stealth_set_mimic_off)
    END_KEYWORD

  END_KEYWORD_SELECT
END DEFINE_TABLE

```

Table 2.5-127: DEFINE_TABLE (stealth_mode)

```
DEFINE_TABLE (stealth_table)
  KEYWORD_SELECT (" Options for the stealth")

    KEYWORD ("set_tick", "- set time in seconds for symbolics drawing of stealth")
      GETDECIMAL ("time in seconds")
      CALL (stealth_set_symbolics_draw_tick)
    END_KEYWORD

    KEYWORD ("address", "- set site host pair for stealth to talk to")
      GETDECIMAL ("site")
      GETDECIMAL ("host")
      CALL (stealth_site_host_pair)
    END_KEYWORD

    KEYWORD ("teleport", "- send stealth to specified x y location")
      GETDECIMAL ("X position")
      GETDECIMAL ("Y position")
      GETDECIMAL ("math degrees")
      CALL (stealth_teleport_to)
    END_KEYWORD

    KEYWORD ("attach_to", "- attach stealth to specified vehicle")
      GETDECIMAL ("vehicle id")
      CALL (stealth_attach_to)
    END_KEYWORD

    KEYWORD ("mimic", "- mode for stealth, attach or mimic(default)")
      DO_KEYWORD_TABLE(stealth_mode)
    END_KEYWORD

  END_KEYWORD_SELECT
END_DEFINE_TABLE
```

Table 2.5-128: DEFINE_TABLE (stealth_table)

```

DEFINE_TABLE (vehicle_table)
  KEYWORD_SELECT (" Options for the vehicle command")

    KEYWORD ("bong", "- deadly vehicle impact")
      CALL (vehicle_bong_do)
    END_KEYWORD

    KEYWORD ("defuel", "- wipe out a vehicle's fuel supply (TANKS ONLY)")
      CALL (vehicle_defuel)
    END_KEYWORD

    KEYWORD ("fake", "- fake resupply a vehicle")
      CALL (vehicle_fake_resupply)
    END_KEYWORD

    KEYWORD ("kill", "- catastrophic vehicle kill")
      CALL (vehicle_catastrophe)
    END_KEYWORD

    KEYWORD ("ping", "- mild vehicle impact")
      CALL (vehicle_ping_do)
    END_KEYWORD

    KEYWORD ("resupply", "- resupply a vehicle")
      GETDECIMAL ("id of vehicle to resupply from")
      CALL (vehicle_resupply)
    END_KEYWORD

    KEYWORD ("show", "- show vehicle status")
      GETFIELDS (show_table)
      CALL (show_vehicle)
    END_KEYWORD

    KEYWORD ("targeting", "- adjust targeting parameters")
      GETFIELDS (firestatus_table)
      GETDECIMAL ("Max engagement range")
      CALL (parser_set_targeting_parameters)
    END_KEYWORD

  END_KEYWORD_SELECT
END_DEFINE_TABLE

```

Table 2.5-129: DEFINE_TABLE (vehicle_table)

```

DEFINE_TABLE (command_table)
  KEYWORD_SELECT (" Commands")

    KEYWORD ("create", "- create an echelon")
      GETFIELDS (create_side_table)
      GETSTRING ("vehicle, platoon, company-reinforced, etc.")
      GETSTRING ("tank, motorized-rifle, mechanized-infantry, etc.")
      GETDECIMAL ("X position")
      GETDECIMAL ("Y position")
      GETSTRING ("formation")
      GETDECIMAL ("workstation port number or 0")
      CALL (parser_create_vehicle)
    END_KEYWORD

```

Table 2.5-130: DEFINE_TABLE (command_table) Part 1 of 2

```
KEYWORD ("debug", "- debugging operations")
  GETFIELDS (debugging_table)
  GET_ON_OR_OFF
  CALL (toggle_debugging)
END_KEYWORD

KEYWORD ("exit", "- exit program")
  CALL (saf_exit)
END_KEYWORD

KEYWORD ("hashing", "- simnet id hashing statistics")
  DO_KEYWORD_TABLE(hashing_table)
END_KEYWORD

KEYWORD ("heap", "- heap operations")
  DO_KEYWORD_TABLE(heap_table)
END_KEYWORD

KEYWORD ("net", "- network commands")
  DO_KEYWORD_TABLE (net_table)
END_KEYWORD

KEYWORD ("print", "- print commands")
  DO_KEYWORD_TABLE (print_table)
END_KEYWORD

KEYWORD ("quit", "- exit program")
  CALL (saf_exit)
END_KEYWORD

KEYWORD ("reset", "- global phantom reset : **USE WITH CAUTION**")
  DO_KEYWORD_TABLE (parser_reset_table)
END_KEYWORD

KEYWORD ("message", "- send a message to the Symbolics screen")
  GETSTRING ("Message to send")
  GETDECIMAL ("Port to send message to")
  CALL (parser_send_string)
END_KEYWORD

KEYWORD ("set", "- set program parameters")
  DO_KEYWORD_TABLE (set_table)
END_KEYWORD

KEYWORD ("stealth", "- stealth operations")
  DO_KEYWORD_TABLE (stealth_table)
END_KEYWORD

KEYWORD ("vehicle", "- vehicle operations")
  GETDECIMAL("vehicle id")
  DO_KEYWORD_TABLE (vehicle_table)
END_KEYWORD

END_KEYWORD_SELECT
END_DEFINE_TABLE
```

Table 2.5-130: DEFINE_TABLE (command_table) Part 2 of 2

2.5.2.2 debug.h CSU

/simnet/src/host/debug.h

This CSU contains the bitfield templates for activating various parts of the debugging printouts while the phantom program is running. In addition to the constant definition tables, the CSU contains a number of macros that are defined in Appendix A.

Constant	Value
D GLOBAL	0xFFFFFFFF
D EVENT	0x00000001
D OBSTACLE	0x00000002
D COLLISION	0x00000004
D GUNNER	0x00000008
D DRIVER	0x00000010
D PILOT /* Analogous to DRIVER */	0x00000010
D NAVIGATOR	0x00000020
D INTERVIS	0x00000040
D TURRET	0x00000080
D WEAPON	0x00000100
D STATION	0x00000200
D VEHICLE	0x00000400
D AIR	0x00001000
D GROUND	0x00002000
D COMPOSITE	0x00008000
D TARGETING	0x00040000
D TACTICAL	0x00100000
D REMOTE	0x00800000
D REPORTER	0x01000000
D MISSILE	0x02000000
D LOADER	0x04000000
D CM	0x08000000

Table 2.5-131: debug.h Debugging Constant Definitions

Constant	Value
S PILOT	0x00000001
S NAVIGATOR	0x00000002
S TARGETING	0x00000004
S VEHICLE	0x00000010
S AIR	0x00000020
S FORMATION	0x00000040
S REMOTE	0x00000080
S INTERVIS	0x00000100
S GUNNER	0x00000200
S SAFVEH	0x00000400
S WEAPON	0x00000800
S COMPOSITE	0x00001000
S STATE UPDATE	0x00002000
S ENTITY	0x00004000
S TURRET	0x00008000
S SAFENT	0x00010000
S TACTICAL	0x00200000
S REPORTER	0x00800000
S DRIVER	0x01000000
S SPOTTER	0x02000000
S DETECTION	0x04000000
S ALL	0xFFFFFFFF
S DEFAULT	0x00014412

Table 2.5-132: debug.h Show Constant Definitions

2.6 LOCAL VEHICLES CSC

The local vehicles are those vehicles which are simulated by the SAF Simhost CSCI. The Local Vehicles CSCI includes code for simulating vehicle maneuver, for determining which enemy vehicles have been seen and detected, for assessing damage from weapons, for tracking supply consumption, and for performing resupply operations. Each time a vehicle is ticked by the scheduler, it simulates its activities during the interval from the last time it ticked to the current time. It processes the packets which are sent to it (such as vehicle impact packets). It also generates an appearance packet which is sent to the SIMNET interface. The packet will be sent out on the SIMNET LAN if the state of the vehicle has changed sufficiently compared to the state of the vehicle's last appearance packet.

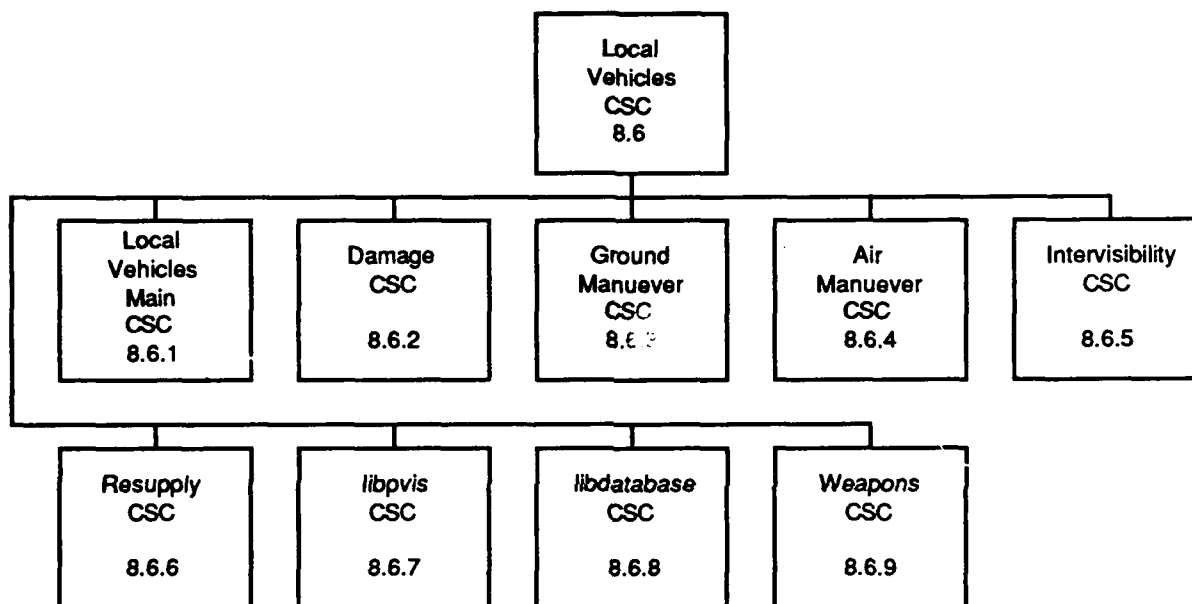


Figure 2.6-1: Local Vehicles CSC Structure

2.6.1 Local Vehicles Main CSC

The Local Vehicles Main CSC [8.6.1] consists of a single file, the saf_vehicle.c CSC [8.6.1.1]. The structure of CSC 8.6.1 is found in the following tables.

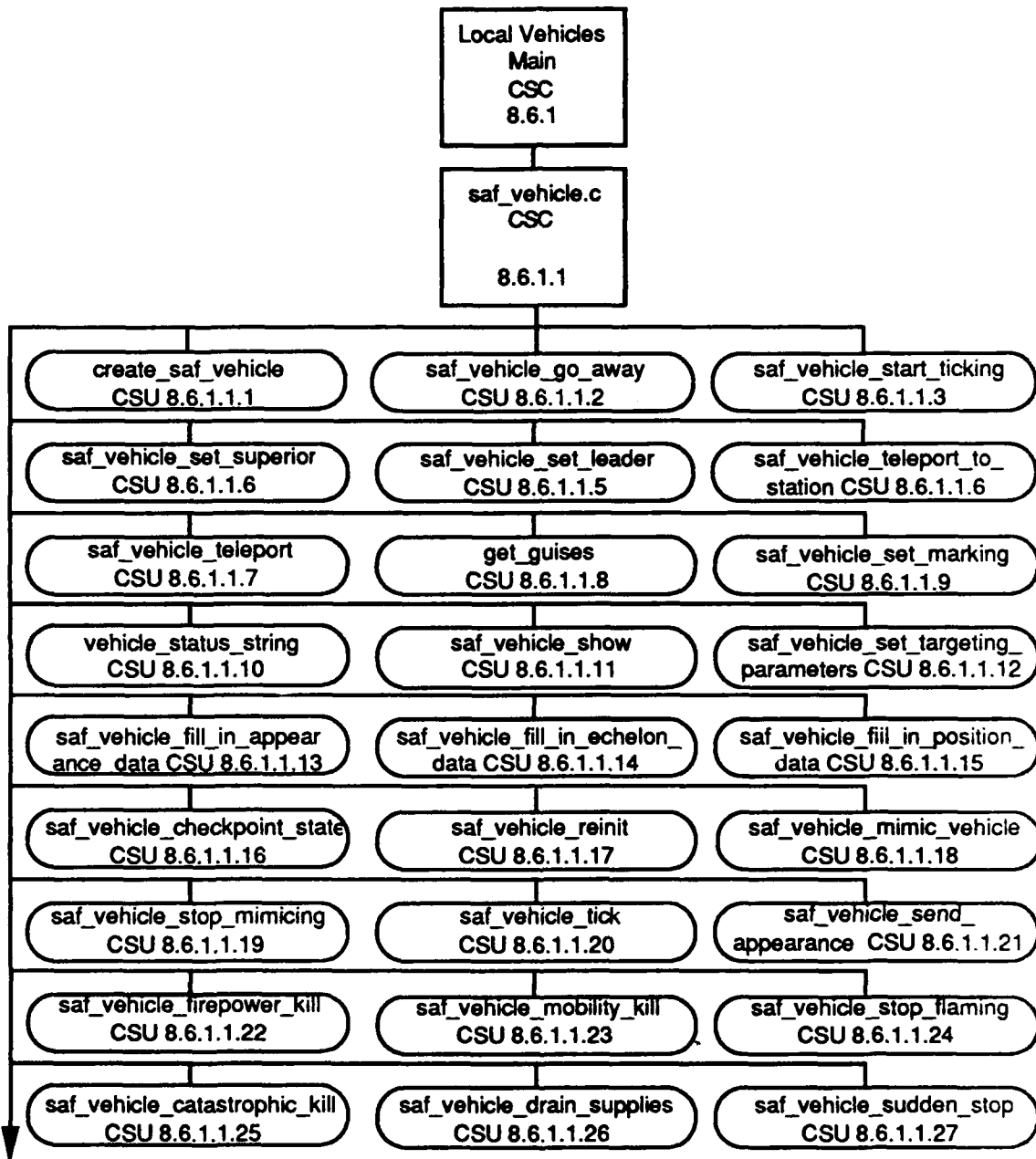


Figure 2.6-2: Local Vehicles Main CSC Structure Part 1 of 2

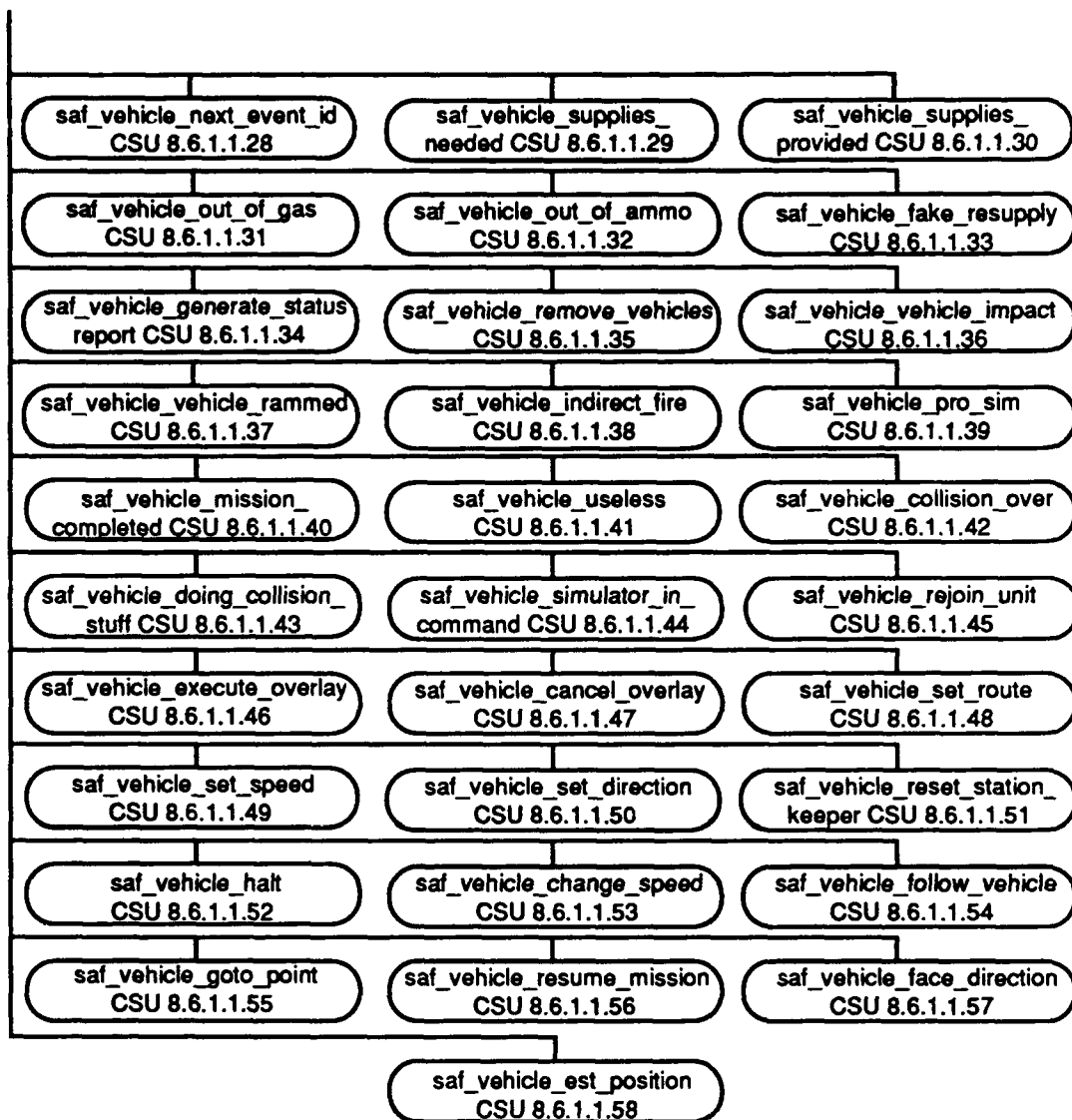


Figure 2.6-3: Local Vehicles Main CSC Structure Part 2 of 2

2.6.1.1 saf_vehicle.c CSC

```
/simnet/src/host/saf_vehicle.c
```

This file contains the CSUs which operate on all saf vehicles, including creation, deletion, state changes, reporting, resupply, mission and immediate commands.

2.6.1.1.1 create_saf_vehicle CSU

This CSU creates a SAF vehicle.

Parameters		
Parameters	Type	Where Typedef Declared
forceID	ForceID	basic.h
countryD	unsigned char	Standard
countryO	unsigned char	Standard
tactics	unsigned char	Standard
*unit_type	pointer to char	Standard
*job	pointer to char	Standard
battalion	unsigned short	Standard
company	unsigned char	Standard
platoon	unsigned char	Standard
bumper	unsigned char	Standard
heading	REAL	sim_types.h
*position	pointer to REAL	sim_types.h
*owner	pointer to SBX CONNECTION VARS	Sec. 2.4.3.3
percent ammo	REAL	sim_types.h
percent fuel	REAL	sim_types.h
ReturnValues		
Return Value	Type	Meaning
safobj	pointer to SAF_OBJECT	Vehicle created.
Calls		
Function	Where Described	
find_tag	Sec. 2.1.1.4.3	
generate vehicle id		
allocate saf vehicle	Sec. 2.9.1.2 See Appendix A	
deg to rad	sim macros.h	
ft float	Sec. 2.14.1.2.12	
heap allocate	Sec. 2.14.2.1.1	
buffer allocate	Sec. 2.14.4.2.12	
get me a random fraction	Sec. 2.14.3.7.1	
allocate safobj	Sec. 2.2.2.2	
create tickable	Sec. 2.2.3.1	
ft int	Sec. 2.14.1.2.11	
create entity	Sec. 2.9.2.1.1	
vec copy	Sec. 2.6.2.59.1 Vehicles CSCI SDD	
create vehicle	Sec. 2.9.2.2.1	
get guises	Sec. 2.6.1.1.8	
saf vehicle set marking	Sec. 2.6.1.1.9	
init grid entry list	Sec. 2.9.3.1.11	
create damage	Sec. 2.6.2.1.1	
create turret	Sec. 2.6.9.6.1	
IS GROUNDVEH		
create groundveh	Sec. 2.6.3.4.1	

Table 2.6-1 is continued on the following page

Function	Where Described
create driver	Sec. 2.6.3.2.1
create collision	Sec. 2.6.3.1.1
IS HELI	
create airveh	Sec. 2.6.4.6.2
create pilot	Sec. 2.6.4.2.2
s_atan2	Sec. 2.14.3.9 See Appendix A
IS PLANE	
create targeting	Sec. 2.6.9.3.2
create weapon systems	Sec. 2.6.9.8.2
create logistics	Sec. 2.6.6.1.1
create detection	Sec. 2.6.5.1.1
saf_vehicle teleport	Sec. 2.6.1.1.7

Table 2.6-1: create_saf_vehicle CSU [8.6.1.1.1]

2.6.1.1.2 saf_vehicle_go_away CSU

This CSU makes a SAF vehicle go away.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
generate a deactivate	Sec. 2.3.1.1	
cancel fncl group	Sec. 2.2.1.1.5	
destroy tickable	Sec. 2.2.3.2	
destroy entity	Sec. 2.9.2.1.2	
destroy grid entry list	Sec. 2.9.3.1.13	
destroy vehicle	Sec. 2.9.2.2.2	
destroy damage	Sec. 2.6.2.1.2	
destroy turret	Sec. 2.6.9.6.2	
destroy groundveh	Sec. 2.6.3.4.2	
destroy airveh	Sec. 2.6.4.6.3	
destroy driver	Sec. 2.6.3.2.2	
destroy collision	Sec. 2.6.3.1.2	
destroy pilot	Sec. 2.6.4.2.4	
destroy targeting	Sec. 2.6.9.3.3	
destroy weapon systems	Sec. 2.6.9.8.3	
destroy logistics	Sec. 2.6.6.1.2	
destroy detection	Sec. 2.6.5.1.3	
buffer deallocate	Sec. 2.14.4.2.15	
deallocate saf_vehicle	Sec. 2.9.1.2 See Appendix A	
deallocate safobj	Sec. 2.2.2.3	
unhash saf_id		

Table 2.6-2: saf_vehicle_go_away CSU [8.6.1.1.2]

2.6.1.1.3 saf_vehicle_start_ticking CSU

This CSU starts a timer.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
start ticking	Sec. 2.2.3.3	

Table 2.6-3: saf_vehicle_start_ticking CSU [8.6.1.1.3]

2.6.1.1.4 saf_vehicle_set_superior CSU

This CSU designates the superior vehicle.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*superior	pointer to SAF_OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
DEBUG COMPOSITE	Sec. 2.5.2.2 See Appendix A	
OBJ_VEHICLEID	Sec. 2.9.1.1 See Appendix A	
composite_remove_inferior_vehicle	Sec. 2.8.1.3.8	
composite_add_inferior_vehicle	Sec. 2.8.1.3.7	

Table 2.6-4: saf_vehicle_set_superior CSU [8.6.1.1.4]

2.6.1.1.5 saf_vehicle_set_leader CSU

This CSU designates the leader.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
id	unsigned int	Standard
xoff	int	Standard
yoff	int	Standard
unit_lead_id	unsigned int	Standard
unit_xoff	int	Standard
unit_yoff	int	Standard
*orderer	pointer to SAF_OBJECT	Sec. 2.9.1.1

Calls	
Function	Where Described
DEBUG VEHICLE	Sec. 2.5.2.2 See Appendix A
OBJ VEHICLEID	Sec. 2.9.1.1 See Appendix A
driver set leader mis	Sec. 2.6.3.2.18
driver follow leader	Sec. 2.6.3.2.19
pilot set leader mis	Sec. 2.6.4.2.79
pilot follow leader	Sec. 2.6.4.2.80

Table 2.6-5: saf_vehicle_set_leader CSU [8.6.1.1.5]

2.6.1.1.6 saf_vehicle_teleport_to_station CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
mission stationpoint	Sec. 2.6.3.2.9	
pilot stationpoint	Sec. 2.6.4.2.78	
saf_vehicle_teleport	Sec. 2.6.1.1.7	

Table 2.6-6: saf_vehicle_teleport_to_station CSU [8.6.1.1.6]

2.6.1.1.7 saf_vehicle_teleport CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
x	REAL	sim_types.h
y	REAL	sim_types.h
Calls		
Function	Where Described	
s atan2	Sec. 2.14.3.9 See Appendix A	
coords within database	Sec. 2.13.3.1 See Appendix A	
tdb place vehicle	Sec. 2.21.7.20.5	
report error from tdb once	Sec. 2.14.1.2.4	
mat rot init	Sec. 2.6.2.47.1 Vehicles CSCI SDD	
update grid entry list	Sec. 2.9.3.1.12	

Table 2.6-7: saf_vehicle_teleport CSU [8.6.1.1.7]

2.6.1.1.8 get_guises CSU

This CSU determines the vehicle guises.

Parameters		
Parameters	Type	Where Typedef Declared
*table	pointer to DATA UNION	Sec. 2.1.1.5
distinguished county	unsigned short	Standard
other country	unsigned short	Standard
*guises	pointer to VehicleGuises	basic.h
Calls		
Function	Where Described	
ft int	Sec. 2.14.1.2.11	

Table 2.6-8: get_guises CSU [8.6.1.1.8]

2.6.1.1.9 saf_vehicle_set_marking CSU

This CSU sets the vehicle markings.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
battalion	unsigned short	Standard
company	unsigned char	Standard
platoon	unsigned char	Standard
bumper	int	Standard

Table 2.6-9: saf_vehicle_set_marking CSU [8.6.1.1.9]

2.6.1.1.10 vehicle_status_string CSU

Within this CSU, the macro ADD_STRING(s,p,b) is defined. The definition is included in Appendix A.

Parameters		
Parameters	Type	Where Typedef Declared
status	int	Standard
ReturnValues		
Return Value	Type	Meaning
buf	pointer to char	Text to be displayed.

Table 2.6-10: vehicle_status_string CSU [8.6.1.1.10]

2.6.1.1.11 saf_vehicle_show CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
flags	int	Standard
Calls		
Function	Where Described	
entity show	Sec. 2.9.2.1.4	
vehicle show	Sec. 2.9.2.2.4	
driver show	Sec. 2.6.3.2.43	
collision show	Sec. 2.6.3.1.4	
airveh show	Sec. 2.6.4.6.4	
pilot show	Sec. 2.6.4.2.7	
turret show	Sec. 2.6.9.6.3	
targeting show	Sec. 2.6.9.3.5	
detection show	Sec. 2.6.5.1.5	
weapon systems show	Sec. 2.6.9.8.5	
OBJ VEHICLEID	Sec. 2.9.1.1 See Appendix A	
vehicle status string	Sec. 2.6.1.1.10	

Table 2.6-11: saf_vehicle_show CSU [8.6.1.1.11]

2.6.1.1.12 saf_vehicle_set_targeting_parameters CSU

This CSU sets the targeting parameters.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
firestatus	int	Standard
max engagement range	REAL	sim_types.h
marksmanship	REAL	sim_types.h
position x	REAL	sim_types.h
position y	REAL	sim_types.h
radius	REAL	sim_types.h
targets[]	unsigned short	Standard
Calls		
Function	Where Described	
targeting_set parameters	Sec. 2.6.9.3.11	

Table 2.6-12: saf_vehicle_set_targeting_parameters CSU [8.6.1.1.12]

2.6.1.1.13 saf_vehicle_fill_in_appearance_data CSU

This CSU fills in the appearance data for a vehicle.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*desc_ptr	pointer to VEHICLE_APPEARANCE_ DESCRIPTOR	Sec. 2.4.1.1
Calls		
Function	Where Described	
entity_fill_in_appearance_data	Sec. 2.9.2.1.6	
vehicle_fill_in_appearance_data	Sec. 2.9.2.2.6	

Table 2.6-13: saf_vehicle_fill_in_appearance_data CSU [8.6.1.1.13]

2.6.1.1.14 saf_vehicle_fill_in_echelon_data CSU

This CSU fills in the vehicle echelon data.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*desc_ptr	pointer to VEHICLE_ECHELON_ DESCRIPTOR	Sec. 2.4.1.1
Calls		
Function	Where Described	
OBJ_VEHICLEID	Sec. 2.9.1.1 See Appendix A	

Table 2.6-14: saf_vehicle_fill_in_echelon_data CSU [8.6.1.1.14]

2.6.1.1.15 saf_vehicle_fill_in_position_data CSU

This CSU fills in the vehicle position data.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*desc_ptr	pointer to VEHICLE_POSITION_ DESCRIPTOR	Sec. 2.4.1.1

Calls	
Function	Where Described
entity fill in position data	Sec. 2.9.2.1.5
vehicle fill in position data	Sec. 2.9.2.2.5

Table 2.6-15: saf_vehicle_fill_in_position_data CSU [8.6.1.1.15]

2.6.1.1.16 saf_vehicle_checkpoint_state CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
port number	int	Standard

Calls	
Function	Where Described
buffer allocate	Sec. 2.14.4.2.12
fill sbx opfor header	Sec. 2.4.3.2.18
OBJ VEHICLEID	Sec. 2.9.1.1 See Appendix A
weapon systems checkpoint	Sec. 2.6.9.8.9
sbx_connection_send_to_port	Sec. 2.4.3.2.19
buffer deallocate	Sec. 2.14.4.2.15

Table 2.6-16: saf_vehicle_checkpoint_state CSU [8.6.1.1.16]

2.6.1.1.17 saf_vehicle_reinit CSU

This CSU reinitializes a SAF vehicle.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
loads[]	short	Standard
fuel	int	Standard
status	char	Standard
x	REAL	sim types.h
y	REAL	sim types.h
bearing	REAL	sim types.h

Calls	
Function	Where Described
weapon systems reinit	Sec. 2.6.9.8.10
saf vehicle catastrophic kill	Sec. 2.6.1.1.25
saf vehicle mobility kill	Sec. 2.6.1.1.23
saf vehicle firepower kill	Sec. 2.6.1.1.22

Table 2.6-17: saf_vehicle_reinit CSU [8.6.1.1.17]

2.6.1.1.18 saf_vehicle_mimic_vehicle CSU

This CSU causes a vehicle to mimic another vehicle.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
vid	unsigned int	Standard
Calls		
Function	Where Described	
simnet send deactivate		
OBJ VEHICLEID	Sec. 2.9.1.1 See Appendix A	
RETYPE VEHICLE	Sec. 2.9.3.2 See Appendix A	

Table 2.6-18: saf_vehicle_mimic_vehicle CSU [8.6.1.1.18]

2.6.1.1.19 saf_vehicle_stop_mimicing CSU

This CSU causes a vehicle to stop mimicing

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
*new_pos	REAL	sim types.h
Calls		
Function	Where Described	
vec_copy	Sec. 2.6.2.59.1 Vehicles CSCI SDD	
RETYPE VEHICLE	Sec. 2.9.3.2 See Appendix A	
OBJ VEHICLEID	Sec. 2.9.1.1 See Appendix A	

Table 2.6-19: saf_vehicle_stop_mimicing CSU [8.6.1.1.19]

2.6.1.1.20 saf_vehicle_tick CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1

Calls	
Function	Where Described
tickable note start tick	Sec. 2.2.3.6
change tick rate	Sec. 2.2.3.5
vec copy	Sec. 2.6.2.59.1 Vehicles CSCI SDD
LOOKUP VEHICLE	Sec. 2.9.3.2 See Appendix A
OBJ POSITION	Sec. 2.9.1.1 See Appendix A
OBJ VELOCITY	Sec. 2.9.1.1 See Appendix A
OBJ HULL TO WORLD	Sec. 2.9.1.1 See Appendix A
OBJ DIRECTION	Sec. 2.9.1.1 See Appendix A
groundveh tick	Sec. 2.6.3.4.4
air tick	Sec. 2.6.4.6.5
saf_vehicle_send_appearance	Sec. 2.6.1.1.21
update_grid_entry_list	Sec. 2.9.3.1.12
buffer_simple_dequeue	Sec. 2.14.4.1.15
buffer_deallocate	Sec. 2.14.4.2.15
saf_vehicle_pro_sim	Sec. 2.6.1.1.39
gunner_round_flying	Sec. 2.6.9.5.6
targeting tick	Sec. 2.6.9.3.21
collision tick	Sec. 2.6.3.1.6
driver tick	Sec. 2.6.3.2.4
pilot tick	Sec. 2.6.4.2.64
collision_dead tick	Sec. 2.6.3.1.7
logistics tick	Sec. 2.6.6.1.10
max	Sec. 2.13.3.5 & Sec. 2.6.7.3 See Appendix A
saf_vehicle_out_of_gas	Sec. 2.6.1.1.31

Table 2.6-20: saf_vehicle_tick CSU [8.6.1.1.20]

2.6.1.1.21 saf_vehicle_send_appearance CSU

This CSU sends an appearance packet over the network.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
simnet_send_appearance		

Table 2.6-21: saf_vehicle_send_appearance CSU [8.6.1.1.21]

2.6.1.1.22 saf_vehicle_firepower_kill CSU

This CSU kills a vehicle's firepower.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
broadcast appearance data	Sec. 2.4.3.2.73	
turret firepower kill	Sec. 2.6.9.6.10	

Table 2.6-22: saf_vehicle_firepower_kill CSU [8.6.1.1.22]

2.6.1.1.23 saf_vehicle_mobility_kill CSU

This CSU renders a vehicle immobile.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
groundveh mobility kill	Sec. 2.6.3.4.5	
saf vehicle catastrophic kill	Sec. 2.6.1.1.25	
broadcast appearance data	Sec. 2.4.3.2.73	
composite note leader state	Sec. 2.8.1.3.36	
composite_inferior_changed_status	Sec. 2.8.1.3.37	

Table 2.6-23: saf_vehicle_mobility_kill CSU [8.6.1.1.23]

2.6.1.1.24 saf_vehicle_stop_flaming CSU

This CSU causes the vehicle to stop burning.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1

Table 2.6-24: saf_vehicle_stop_flaming CSU [8.6.1.1.24]

2.6.1.1.25 saf_vehicle_catastrophic_kill CSU

This CSU destroys a vehicle.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
groundveh_mobility_kill	Sec. 2.6.3.4.5	
deferred_incl	Sec. 2.2.1.1.1	
airveh_catastrophic_kill	Sec. 2.6.4.6.7	
broadcast_appearance_data	Sec. 2.4.3.2.73	
composite_note_leader_state	Sec. 2.8.1.3.36	
composite_inferior_changed_status	Sec. 2.8.1.3.37	
composite_note_member_vehicle_has_died	Sec. 2.8.1.3.38	

Table 2.6-25: saf_vehicle_catastrophic_kill CSU [8.6.1.1.25]

2.6.1.1.26 saf_vehicle_drain_supplies CSU

This CSU decrements the vehicle's supplies.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1

Table 2.6-26: saf_vehicle_drain_supplies CSU [8.6.1.1.26]

2.6.1.1.27 saf_vehicle_sudden_stop CSU

This CSU causes a vehicle to stop suddenly.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1

Table 2.6-27: saf_vehicle_sudden_stop CSU [8.6.1.1.27]

2.6.1.1.28 saf_vehicle_next_event_id CSU

This CSU returns the id for the next event.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1

ReturnValues		
Return Value	Type	Meaning
safobj->vehicle->network_event	int	Next event id.

Table 2.6-28: saf_vehicle_next_event_id CSU [8.6.1.1.28]

2.6.1.1.29 saf_vehicle_supplies_needed CSU

This CSU determines if a vehicle is out of fuel.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*num_ptr	pointer to int	Standard
*sup_ptr	pointer to MunitionQuantity	basic.h

Table 2.6-29: saf_vehicle_supplies_needed CSU [8.6.1.1.29]

2.6.1.1.30 saf_vehicle_supplies_provided CSU

This CSU resupplies a vehicle.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
num	int	Standard
*sup_ptr	pointer to MunitionQuantity	basic.h
Calls		
Function	Where Described	
composite_note_leader_state	Sec. 2.8.1.3.36	
composite_inferior_changed_status	Sec. 2.8.1.3.37	

Table 2.6-30: saf_vehicle_supplies_provided CSU [8.6.1.1.30]

2.6.1.1.31 saf_vehicle_out_of_gas CSU

This CSU causes a vehicle to run out of gas.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1

Calls	
Function	Where Described
DEBUG VEHICLE	Sec. 2.5.2.2 See Appendix A
composite_note_leader_state	Sec. 2.8.1.3.36
composite_inferior_changed_status	Sec. 2.8.1.3.37

Table 2.6-31: saf_vehicle_out_of_gas CSU [8.6.1.1.31]

2.6.1.1.32 saf_vehicle_out_of_ammo CSU

This CSU causes a vehicle to run out of ammunition.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
DEBUG VEHICLE	Sec. 2.5.2.2 See Appendix A	

Table 2.6-32: saf_vehicle_out_of_ammo CSU [8.6.1.1.32]

2.6.1.1.33 saf_vehicle_fake_resupply CSU

This CSU fakes a vehicle resupply.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
composite_note_leader_state	Sec. 2.8.1.3.36	
composite_inferior_changed_status	Sec. 2.8.1.3.37	
DEBUG VEHICLE	Sec. 2.5.2.2 See Appendix A	
weapon systems rearm	Sec. 2.6.9.8.8	

Table 2.6-33: saf_vehicle_fake_resupply CSU [8.6.1.1.33]

2.6.1.1.34 saf_vehicle_generate_status_report CSU

This CSU prints out a status report about a SAF vehicle.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1

Calls	
Function	Where Described
buffer allocate	Sec. 2.14.4.2.12
vec_mag3	sim_macros.h
OBJ_VELOCITY	Sec. 2.9.1.1 See Appendix A
OBJ_POSITION	Sec. 2.9.1.1 See Appendix A
tdb_giv_xy_get_utm	Sec. 2.21.7.24.3
LOOKUP_VEHICLE	Sec. 2.9.3.2 See Appendix A
sbx_printf	Sec. 2.4.3.2.8
OBJ_VEHICLEID	Sec. 2.9.1.1 See Appendix A
OBJ_OWNER_PORT_NUMBER	Sec. 2.9.1.1 See Appendix A
clear weapons status	Sec. 2.6.9.8.11
add weapons status	Sec. 2.6.9.8.12
fill sbx_opfor_header	Sec. 2.4.3.2.18
driver_executing_immediate_command	Sec. 2.6.3.2.15
pilot_executing_immediate_command	Sec. 2.6.4.2.72
mps to kph	Sec. 2.13.3.1 See Appendix A
mps to knots	Sec. 2.13.3.1 See Appendix A
vehicle status string	Sec. 2.6.1.1.10
sbx_connection_send_to_port	Sec. 2.4.3.2.19
buffer_deallocate	Sec. 2.14.4.2.15

Table 2.6-34: saf_vehicle_generate_status_report CSU [8.6.1.1.34]

2.6.1.1.35 saf_vehicle_remove_vehicles CSU

This CSU removes the designated number of vehicles from the active list.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
num	int	Standard
v_list[]	unsigned int	Standard

Calls	
Function	Where Described
driver_remove_vehicles	Sec. 2.6.3.2.3
pilot_remove_vehicles	Sec. 2.6.4.2.5
collision_remove_vehicles	Sec. 2.6.3.1.3
logistics_remove_vehicles	Sec. 2.6.6.1.3
detection_remove_vehicles	Sec. 2.6.5.1.4

Table 2.6-35: saf_vehicle_remove_vehicles CSU [8.6.1.1.35]

2.6.1.1.36 saf_vehicle_vehicle_impact CSU

This CSU causes a vehicle to be impacted by a round.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*spdu	pointer to SimulationPDU	p_sim.h
Calls		
Function	Where Described	
damage_vehicle_impact	Sec. 2.6.2.1.4	

Table 2.6-36: saf_vehicle_vehicle_impact CSU [8.6.1.1.36]

2.6.1.1.37 saf_vehicle_vehicle_rammed CSU

This CSU causes two vehicles to collide.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*spdu	pointer to SimulationPDU	p_sim.h
Calls		
Function	Where Described	
damage_vehicle_rammed	Sec. 2.6.2.1.5	
collision_vehicle_rammed	Sec. 2.6.3.1.12	

Table 2.6-37: saf_vehicle_vehicle_rammed CSU [8.6.1.1.37]

2.6.1.1.38 saf_vehicle_indirect_fire CSU

This CSU causes a vehicle to be hit by indirect fire.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*spdu	pointer to SimulationPDU	p_sim.h
Calls		
Function	Where Described	
damage_indirect_fire	Sec. 2.6.2.1.6	

Table 2.6-38: saf_vehicle_indirect_fire CSU [8.6.1.1.38]

2.6.1.1.39 saf_vehicle_pro_sim CSU

This CSU generates protocol data units.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*spdu	pointer to SimulationPDU	p_sim.h
Errors		
Error Name	Reason for Error	
Bad packet type ... for ...	Unknown packet type.	
Calls		
Function	Where Described	
saf_vehicle_vehicle impact	Sec. 2.6.1.1.36	
saf_vehicle_indirect fire	Sec. 2.6.1.1.38	
saf_vehicle_vehicle rammed	Sec. 2.6.1.1.37	
logistics_supply_offer_received	Sec. 2.6.6.1.9	
logistics_supply_offer_canceled	Sec. 2.6.6.1.8	
ERROR_OUT	Sec. 2.5.2.2 See Appendix A	
OBJ_VEHICLEID	Sec. 2.9.1.1 See Appendix A	

Table 2.6-39: saf_vehicle_pro_sim CSU [8.6.1.1.39]

2.6.1.1.40 saf_vehicle_mission_completed CSU

This CSU sets the status of a mission to be complete.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
ReturnValues		
Return Value	Type	Meaning
driver_mission_completed(...)	int	Response if driver.
pilot_mission_completed(...)	int	Response if pilot.
TRUE	int	Mission completed.
Calls		
Function	Where Described	
driver_mission_completed	Sec. 2.6.3.2.16	
pilot_mission_completed	Sec. 2.6.4.2.77	

Table 2.6-40: saf_vehicle_mission_completed CSU [8.6.1.1.40]

2.6.1.1.41 saf_vehicle_useless CSU

This CSU returns TRUE if a vehicle is unavailable and returns FALSE if it is available.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*whos_asking	pointer to SAF_OBJECT	Sec. 2.9.1.1
ReturnValues		
Return Value	Type	Meaning
TRUE	int	Vehicle is not available for use.
FALSE	int	Vehicle is available for use.
(saf_vehicle->superior != whos_asking) && composite_executing_order(saf_vehicle->superior)	int	Vehicle is not available if it is not the company asking and the platoon is busy.
Calls		
Function	Where Described	
driver_executing_immediate_command	Sec. 2.6.3.2.15	
pilot_executing_immediate_command	Sec. 2.6.4.2.72	
composite_executing_order	Sec. 2.8.1.3.35	

Table 2.6-41: saf_vehicle_useless CSU [8.6.1.1.41]

2.6.1.1.42 saf_vehicle_collision_over CSU

This CSU ends a collision.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*box	pointer to RECT	Sec. 2.
Calls		
Function	Where Described	
driver_resume_from_collision	Sec. 2.6.3.2.26	

Table 2.6-42: saf_vehicle_collision_over CSU [8.6.1.1.42]

2.6.1.1.43 saf_vehicle_doing_collision_stuff CSU

This CSU disengages a vehicle from a collision.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1

ReturnValues		
Return Value	Type	Meaning
collision disengaging(...)	int	Collision status.
Calls		
Function	Where Described	
collision disengaging	Sec. 2.6.3.1.5	

Table 2.6-43: saf_vehicle_doing_collision_stuff CSU [8.6.1.1.43]

2.6.1.1.44 saf_vehicle_simulator_in_command CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
ReturnValues		
Return Value	Type	Meaning
(saf_object->driver && driver_simulator in command(...))	int	
Calls		
Function	Where Described	
driver simulator in command	Sec. 2.6.3.2.12	

Table 2.6-44: saf_vehicle_simulator_in_command CSU [8.6.1.1.44]

2.6.1.1.45 saf_vehicle_rejoin_unit CSU

This CSU causes a vehicle to rejoin its unit.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
saf_vehicle_cancel_overlay	Sec. 2.6.1.1.47	

Table 2.6-45: saf_vehicle_rejoin_unit CSU [8.6.1.1.45]

2.6.1.1.46 saf_vehicle_execute_overlay CSU

This CSU causes a vehicle to execute the designated overlay.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*overlay	pointer to OVERLAY	Sec. 2.10.2.2
*cm	pointer to CONTROL MEASURE	Sec. 2.10.2.2
Calls		
Function	Where Described	
composite note leader state	Sec. 2.8.1.3.36	
driver execute overlay	Sec. 2.6.3.2.17	
pilot execute overlay	Sec. 2.6.4.2.85	
composite_inferior_changed_status	Sec. 2.8.1.3.37	

Table 2.6-46: saf_vehicle_execute_overlay CSU [8.6.1.1.46]

2.6.1.1.47 saf_vehicle_cancel_overlay CSU

This CSU cancels an overlay.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
composite note leader state	Sec. 2.8.1.3.36	
driver stop mission	Sec. 2.6.3.2.20	
pilot stop mission	Sec. 2.6.4.2.82	
composite_inferior_changed_status	Sec. 2.8.1.3.37	

Table 2.6-47: saf_vehicle_cancel_overlay CSU [8.6.1.1.47]

2.6.1.1.48 saf_vehicle_set_route CSU

This CSU sets the route for the vehicles to follow.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*route	pointer to ROUTE	Sec. 2.10.2.5
*routept	pointer to ROUTEPOINT	Sec. 2.10.2.5
*orderer	pointer to SAF_OBJECT	Sec. 2.9.1.1

Errors	
Error Name	Reason for Error
ERROR_ABORT	No pilot or driver for vehicle.
Calls	
Function	Where Described
driver set route	Sec. 2.6.3.2.22
pilot set route	Sec. 2.6.4.2.81

Table 2.6-48: saf_vehicle_set_route CSU [8.6.1.1.48]

2.6.1.1.49 saf_vehicle_set_speed CSU

This CSU sets the vehicle speed.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
speed	REAL	sim_types.h
ReturnValues		
Return Value	Type	Meaning
driver set speed	int	Speed set by driver.
pilot set speed	int	Speed set by pilot
FALSE	int	There is no driver or pilot.
Calls		
Function	Where Described	
driver set speed	Sec. 2.6.3.2.23	
pilot set speed	Sec. 2.6.4.2.83	

Table 2.6-49: saf_vehicle_set_speed CSU [8.6.1.1.49]

2.6.1.1.50 saf_vehicle_set_direction CSU

This CSU sets the vehicle direction.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
direction	int	Standard
Calls		
Function	Where Described	
driver set direction	Sec. 2.6.3.2.24	

Table 2.6-50: saf_vehicle_set_direction CSU [8.6.1.1.50]

2.6.1.1.51 saf_vehicle_reset_station_keeper CSU

This CSU resets the station keeper.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
driver forget about forps	Sec. 2.6.3.2.21	
p follower set follow	Sec. 2.6.4.3.10	

Table 2.6-51: saf_vehicle_reset_station_keeper CSU [8.6.1.1.51]

2.6.1.1.52 saf_vehicle_halt CSU

This CSU causes a vehicle to stop.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*orderer	pointer to SAF_OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
composite note leader state	Sec. 2.8.1.3.36	
driver halt cmd	Sec. 2.6.3.2.27	
composite_inferior_changed_status	Sec. 2.8.1.3.37	

Table 2.6-52: saf_vehicle_halt CSU [8.6.1.1.52]

2.6.1.1.53 saf_vehicle_change_speed CSU

This CSU changes the speed of a vehicle.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
speed	REAL	sim_types.h
*orderer	pointer to SAF_OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
driver change speed cmd	Sec. 2.6.3.2.33	
pilot change speed in	Sec. 2.6.4.2.68	

Table 2.6-53: saf_vehicle_change_speed CSU [8.6.1.1.53]

2.6.1.1.54 saf_vehicle_follow_vehicle CSU

This CSU causes a vehicle to follow a designated vehicle.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
vid	unsigned int	Standard
xoff	int	Standard
yoff	int	Standard
*orderer	pointer to SAF OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
composite note leader state	Sec. 2.8.1.3.36	
driver follow vehicle cmd	Sec. 2.6.3.2.28	
pilot follow vehicle im	Sec. 2.6.4.2.73	
composite_inferior_changed_status	Sec. 2.8.1.3.37	

Table 2.6-54: saf_vehicle_follow_vehicle CSU [8.6.1.1.54]

2.6.1.1.55 saf_vehicle_goto_point CSU

This CSU causes a vehicle to go to a designated point.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
x	REAL	sim types.h
y	REAL	sim types.h
backwardp	int	Standard
*orderer	pointer to SAF OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
composite note leader state	Sec. 2.8.1.3.36	
driver goto point cmd	Sec. 2.6.3.2.29	
pilot goto point im	Sec. 2.6.4.2.69	
composite_inferior_changed_status	Sec. 2.8.1.3.37	

Table 2.6-55: saf_vehicle_goto_point CSU [8.6.1.1.55]

2.6.1.1.56 saf_vehicle_resume_mission CSU

This CSU causes a mission to be resumed.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*orderer	pointer to SAF_OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
composite_note_leader_state	Sec. 2.8.1.3.36	
driver_resume_mission_cmd	Sec. 2.6.3.2.31	
pilot_cancel_immediate	Sec. 6.4.2.71	
composite_inferior_changed_status	Sec. 2.8.1.3.37	

Table 2.6-56: saf_vehicle_resume_mission CSU [8.6.1.1.56]

2.6.1.1.57 saf_vehicle_face_direction CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
mathradians	REAL	sim_types.h
*orderer	pointer to SAF_OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
composite_note_leader_state	Sec. 2.8.1.3.36	
driver_face_direction_cmd	Sec. 2.6.3.2.30	
pilot_face_direction_im	Sec. 6.4.2.74	
composite_inferior_changed_status	Sec. 2.8.1.3.37	

Table 2.6-57: saf_vehicle_face_direction CSU [8.6.1.1.57]

2.6.1.1.58 saf_vehicle_est_position CSU

This CSU estimates vehicle location when *time* occurs.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
time	unsigned	Standard
location	VECTOR	sim_types.h

Calls	
Function	Where Described
vec scale	Sec. 2.6.2.64.1 Vehicles CSCI SDD
OBJ VELOCITY	Sec. 2.9.1.1 See Appendix A
vec add	Sec. 2.6.2.57.1 Vehicles CSCI SDD
OBJ POSITION	Sec. 2.9.1.1 See Appendix A

Table 2.6-58: saf_vehicle_est_position CSU [8.6.1.1.58]

2.6.2 Damage CSC

The Damage CSC [8.6.2] consists of a single file, the damage.c CSC [8.6.2.1]. The structure of CSC 8.6.2 is found in the following table.

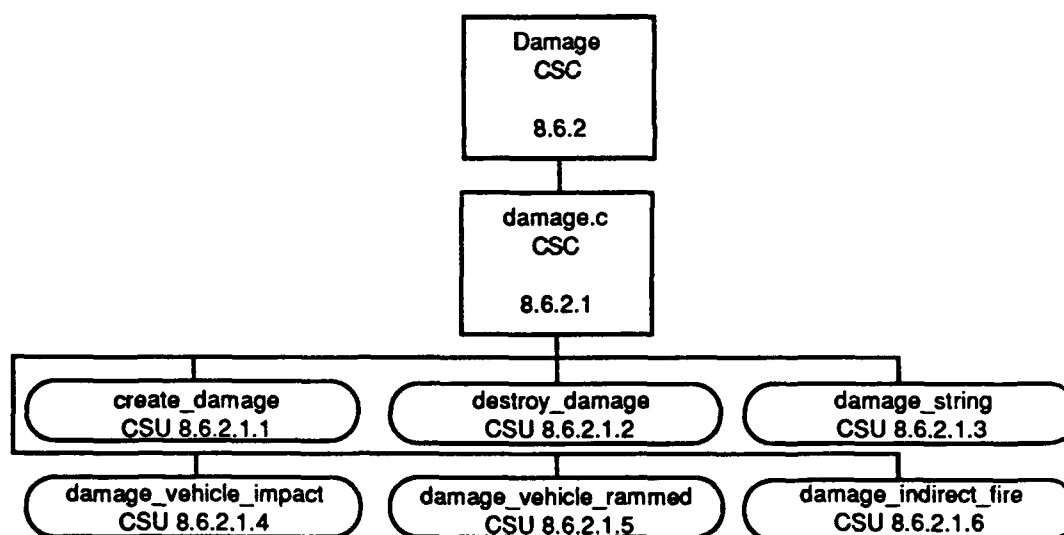


Figure 2.6-4: Damage CSC Structure

2.6.2.1 damage.c CSC

/simnet/src/host/damage.c

This CSC contains the code which handles the damage calculations for vehicles.

The square of the maximum damage range is defined as a constant of 4900 (#define MAX_DAMAGE_RANGE2 4900 /* 70 meters */).

2.6.2.1.1 create_damage CSU

This CSU creates damage.

Parameters		
Parameters	Type	Where Typedef Declared
id	unsigned int	Standard
*table	pointer to DATA_UNION	Sec. 2.1.1.5
ReturnValues		
Return Value	Type	Meaning
damage	pointer to DAMAGE_VARS	Damage created.
Calls		
Function	Where Described	
allocate_damage	Sec. 2.9.1.2 See Appendix A	
ft symbol	Sec. 2.14.1.2.13	
find tag	Sec. 2.1.1.4.3	
ft float	Sec. 2.14.1.2.12	

Table 2.6-59: create_damage CSU [8.6.2.1.1]

2.6.2.1.2 destroy_damage CSU

This CSU calls deallocate_damage.

Parameters		
Parameters	Type	Where Typedef Declared
*damage	pointer to DAMAGE_VARS	Sec. 2.
Calls		
Function	Where Described	
deallocate_damage	Sec. 2.9.1.2 See Appendix A	

Table 2.6-60: destroy_damage CSU [8.6.2.1.2]

2.6.2.1.3 damage_string CSU

This CSU returns a string which describes the damage sustained by a vehicle.

Parameters		
Parameters	Type	Where Typedef Declared
d	int	Standard

ReturnValues		
Return Value	Type	Meaning
"no damage"	pointer to char	No damage.
"catastrophic damage"	pointer to char	Catastrophic damage.
"mobility damage"	pointer to char	Mobility damage.
"firepower damage"	pointer to char	Firepower damage.
"unknown damage"	pointer to char	Damage unknown.

Table 2.6-61: damage_string CSU [8.6.2.1.3]

2.6.2.1.4 damage_vehicle_impact CSU

This CSU determines the damage to a vehicle caused by an impact.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*spdu	pointer to SimulationPDU	p_sim.h
Calls		
Function	Where Described	
get me a random fraction	Sec. 2.14.3.7.1	
saf id from simnet id		
predicate impact hook	Sec. 2.10.1.3.7	
database of damage query	Sec. 2.6.8.6.10	
DEBUG VEHICLE	Sec. 2.5.2.2 See Appendix A	
damage string	Sec. 2.6.2.1.3	
IS HELI		
IS PLANE		
saf vehicle catastrophic kill	Sec. 2.6.1.1.25	
simnet send status change		
saf vehicle mobility kill	Sec. 2.6.1.1.23	
saf vehicle firepower kill	Sec. 2.6.1.1.22	

Table 2.6-62: damage_vehicle_impact CSU [8.6.2.1.4]

2.6.2.1.5 damage_vehicle_rammed CSU

This CSU determines the damage to a vehicle that was involved in a collision.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*spdu	pointer to SimulationPDU	p_sim.h

Calls	
Function	Where Described
OBJ VEHICLEID	Sec. 2.9.1.1 See Appendix A
saf id from simnet id	
get me a random fraction	Sec. 2.14.3.7.1
DEBUG VEHICLE	Sec. 2.5.2.2 See Appendix A
IS HELI	
IS PLANE	
OBJ VEHICLESTATUS	Sec. 2.9.1.1 See Appendix A
saf vehicle mobility kill	Sec. 2.6.1.1.23
simnet send status change	

Table 2.6-63: damage_vehicle_rammed CSU [8.6.2.1.5]

2.6.2.1.6 damage_indirect_fire CSU

This CSU determines the damage from indirect fire.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
*spdu	pointer to SimulationPDU	p_sim.h
Calls		
Function	Where Described	
OBJ VEHICLEID	Sec. 2.9.1.1 See Appendix A	
OBJ POSITION	Sec. 2.9.1.1 See Appendix A	
OBJ VEHICLESTATUS	Sec. 2.9.1.1 See Appendix A	
saf id from simnet id		
range_squared	Sec. 2.14.3.5.10	
get me a random fraction	Sec. 2.14.3.7.1	
database if damage query	Sec. 2.6.8.7.6	
DEBUG VEHICLE	Sec. 2.5.2.2 See Appendix A	
saf vehicle catastrophic kill	Sec. 2.6.1.1.25	
simnet send status change		
saf vehicle mobility kill	Sec. 2.6.1.1.23	
saf vehicle firepower kill	Sec. 2.6.1.1.22	

Table 2.6-64: damage_indirect_fire CSU [8.6.2.1.6]

2.6.3 Ground Maneuver CSC

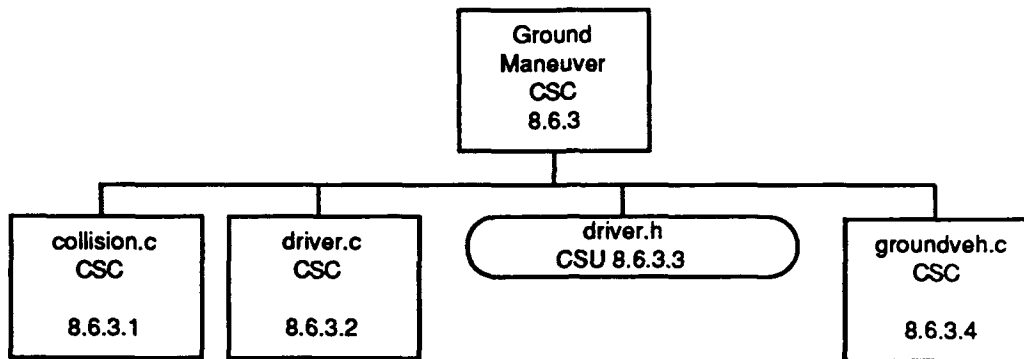


Figure 2.6-5: Ground Maneuver CSC Structure

2.6.3.1 collision.c CSC

/simnet/src/host/collision.c

This CSC contains the code which handles collisions for vehicles.

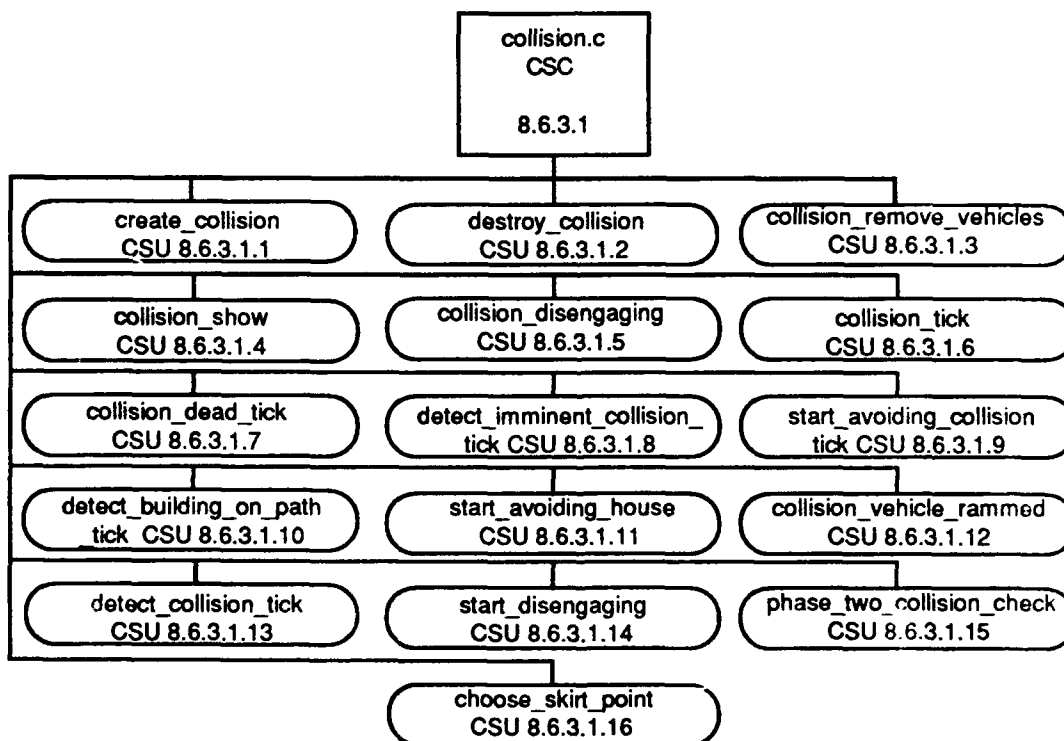


Figure 2.6-6: collision.c CSC Structure

In addition, the file contains the following constant definitions.

Constant	Value
COLLISION STATE WATCHING	1
COLLISION STATE AVOIDING VEHICLE	2
COLLISION STATE AVOIDING HOUSE	3
COLLIISION STATE DISENGAGING	4
COLLISION DISTANCE	6.0
COLLISION DISTANCE WITH LEEWAY	7.0
DISENGAGE TIME	3.0
DISENGAGE MPS	-7.0
SCOOT MPS	2.0
AVOIDANCE DISTANCE	100.0
CRITICAL TIME SECS	4.0
HOUSE LOOKAHEAD TIME	5.0
OBSTACE EXPANSION	3 * INT */
SKIRT DISTANCE	6.0

Table 2.6-x: collision.c Constant Definitions

2.6.3.1.1 create_collision CSU

This CSU allocates memory for and initializes a COLLISION_VARS structure.

Parameters		
Parameters	Type	Where Typedef Declared
*collision	pointer to COLLISION_VARS	Sec. 2.9.1.2
ReturnValues		
Return Value	Type	Meaning
collision	pointer to COLLISION_VARS	Created collision.
Calls		
Function	Where Described	
allocate_collision	Sec. 2.9.1.2 See Appendix A	

Table 2.6-65: create_collision CSU [8.6.3.1.1]

2.6.3.1.2 destroy_collision CSU

This CSU destroys a collision through a call to CSU deallocate_collision.

Parameters		
Parameters	Type	Where Typedef Declared
*collision	pointer to COLLISION_VARS	Sec. 2.9.1.2

Calls	
Function	Where Described
deallocate collision	Sec. 2.9.1.2 See Appendix A

Table 2.6-66: destroy_collision CSU [8.6.3.1.2]

2.6.3.1.3 collision_remove_vehicles CSU

This CSU causes a collision to be removed.

Parameters		
Parameters	Type	Where Typedef Declared
*collision	pointer to COLLISION_VARS	Sec. 2.9.1.2
num	int	Standard
v_list[]	unsigned int	Standard
Calls		
Function	Where Described	
LOOKUP SAFOBJ	Sec. 2.9.1.1 See Appendix A	

Table 2.6-67: collision_remove_vehicles CSU [8.6.3.1.3]

2.6.3.1.4 collision_show CSU

This CSU causes the vehicle to avoid a collision.

Parameters		
Parameters	Type	Where Typedef Declared
*collision	pointer to COLLISION_VARS	Sec. 2.9.1.2
flags	int	Standard
Calls		
Function	Where Described	
OBJ VEHICLEID	Sec. 2.9.1.1 See Appendix A	

Table 2.6-68: collision_show CSU [8.6.3.1.4]

2.6.3.1.5 collision_disengaging CSU

This CSU sets the collision state to COLLISION_STATE_DISENGAGING.

Parameters		
Parameters	Type	Where Typedef Declared
*collision	pointer to COLLISION_VARS	Sec. 2.9.1.2

ReturnValues		
Return Value	Type	Meaning
collision->state == COLLISION_STATE_ DISENGAGING	int	Collision state is set to the value for disengaging.

Table 2.6-69: collision_disengaging CSU [8.6.3.1.5]

2.6.3.1.6 collision_tick CSU

This CSU processes collisions for vehicles which are mobile. It returns FALSE if no collision has occurred. It returns a pointer to the collision state otherwise.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
ReturnValues		
Return Value	Type	Meaning
FALSE	int	No collision.
collision->state != COLLISION_STATE_ WATCHING	int	No collision if collision state is set to the value for watching. Collision otherwise.
Calls		
Function	Where Described	
LOOKUP_VEHICLE	Sec. 2.9.3.2 See Appendix A	
OBJ_FORCEID	Sec. 2.9.1.1 See Appendix A	
saf_vehicle_simulator_in_com mand	Sec. 2.6.1.1.44	
sbx_printf	Sec. 2.4.3.2.8	
OBJ_VEHICLEID	Sec. 2.9.1.1 See Appendix A	
OBJ_OWNER_PORT_NUMB ER	Sec. 2.9.1.1 See Appendix A	
saf_vehicle_sudden_stop	Sec. 2.6.1.1.27	
composite_halt	Sec. 2.8.1.3.43	
saf_vehicle_halt	Sec. 2.6.1.1.52	
DEBUG_COLLISION	Sec. 2.5.2.2 See Appendix A	
OBJ_POSITION	Sec. 2.9.1.1 See Appendix A	
saf_vehicle_collision_over	Sec. 2.6.1.1.42	
detect_collision_tick	Sec. 2.6.3.1.13	
start_disengaging	Sec. 2.6.3.1.14	
detect_building_on_path_tick	Sec. 2.6.3.1.10	
start_avoiding_house	Sec. 2.6.3.1.11	
detect_imminent_collision_ tick	Sec. 2.6.3.1.8	
start_avoiding_collision	Sec. 2.6.3.1.9	
LOOKUP_POSITION	Sec. 2.9.1.1 See Appendix A	
FARTHER3	Sec. 2.14.3.9 See Appendix A	

Table 2.6-70: collision_tick CSU [8.6.3.1.6]

2.6.3.1.7 collision_dead_tick CSU

This CSU processes collisions for vehicles which cannot move. If it detects a collision, it reports it. It does not report it again unless the driver actually disengages and re-collides.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
OBJ POSITION	Sec. 2.9.1.1 See Appendix A	
LOOKUP POSITION	Sec. 2.9.1.1 See Appendix A	
FARTHER3	Sec. 2.14.3.9 See Appendix A	
DEBUG COLLISION	Sec. 2.5.2.2 See Appendix A	
OBJ VEHICLEID	Sec. 2.9.1.1 See Appendix A	
detect collision tick	Sec. 2.6.3.1.13	

Table 2.6-71: collision_dead_tick CSU [8.6.3.1.7]

2.6.3.1.8 detect_imminent_collision_tick CSU

This CSU looks for an imminent collision and, if it finds one, sets the appropriate variables in *collision* and returns TRUE.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
*collision	pointer to COLLISION VARS	Sec. 2.9.1.2
*entity	pointer to ENTITY VARS	Sec. 2.9.1.2
ReturnValues		
Return Value	Type	Meaning
TRUE	int	Collision is imminent.
FALSE	int	Cossision is not imminent.
Calls		
Function	Where Described	
FOR VEHICLES_WITHIN_N_GRIDS DO	Sec. 2.9.3.2 See Appendix A	
OBJ POSITION	Sec. 2.9.1.1 See Appendix A	
OBJ VELOCITY	Sec. 2.9.1.1 See Appendix A	
FARTHER2	Sec. 2.14.3.9 See Appendix A	
phase two collision check	Sec. 2.6.3.1.15	
DEBUG COLLISION	Sec. 2.5.2.2 See Appendix A	
vec copy		
OBJ VEHICLEID	Sec. 2.9.1.1 See Appendix A	

Table 2.6-72: detect_imminent_collision_tick CSU [8.6.3.1.8]

2.6.3.1.9 start_avoiding_collision CSU

This CSU chooses a speed and direction to avoid a collision.

Parameters		
Parameters	Type	Where Typedef Declared
*collision	pointer to COLLISION_VARS	Sec. 2.9.1.2
*saf_vehicle	pointer to SAF_VEHICLE_VARS	Sec. 2.9.1.2
*entity	pointer to ENTITY_VARS	Sec. 2.9.1.2
Calls		
Function	Where Described	
interior_angle_between_vectors	Sec. 2.14.3.5.8	
almost_eq	Sec. 2.14.3.9 See Appendix A	
VEC_LONGER2	Sec. 2.14.3.9 See Appendix A	
which_side	Sec. 2.14.3.5.9	
vec_z_rotate	Sec. 2.	

Table 2.6-73: start_avoiding_collision CSU [8.6.3.1.9]

2.6.3.1.10 detect_building_on_path_tick CSU

This CSU looks for a building on the vehicle's path and, if it finds one, sets the appropriate variables in *collision* and returns TRUE.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*collision	pointer to COLLISION_VARS	Sec. 2.9.1.2
*saf_vehicle	pointer to SAF_VEHICLE_VARS	Sec. 2.9.1.2
*entity	pointer to ENTITY_VARS	Sec. 2.9.1.2
ReturnValues		
Return Value	Type	Meaning
TRUE	int	Building is on path.
FALSE	int	No building is on path.
Calls		
Function	Where Described	
buildings_thru	Sec. 2.12.1.17.1	
DEBUG_COLLISION	Sec. 2.5.2.2	
OBJ_VEHICLEID	Sec. 2.9.1.1	

Table 2.6-74: detect_building_on_path_tick CSU [8.6.3.1.10]

2.6.3.1.11 start_avoiding_house CSU

This CSU chooses a speed and direction to avoid a house.

Parameters		
Parameters	Type	Where Typedef Declared
*collision	pointer to COLLISION VARS	Sec. 2.9.1.2
*saf_vehicle	pointer to SAF VEHICLE VARS	Sec. 2.9.1.2
*entity	pointer to ENTITY VARS	Sec. 2.9.1.2
Calls		
Function	Where Described	
choose skirt point	Sec. 2.6.3.1.16	
DEBUG COLLISION	Sec. 2.5.2.2 See Appendix A	
vec_sub	Sec. 2.6.2.65.1 Vehicles CSCI SDD	
vec_normalize	Sec. 2.6.2.63.1 Vehicles CSCI SDD	

Table 2.6-75: start_avoiding_house CSU [8.6.3.1.11]

2.6.3.1.12 collision_vehicle_rammed CSU

This CSU accepts a report that a collision occurred and notes the information for the next tick.

Parameters		
Parameters	Type	Where Typedef Declared
*collision	pointer to COLLISION VARS	Sec. 2.9.1.2
*spdu	pointer to SimulationPDU	p_sim.h
Calls		
Function	Where Described	
saf_id from simnet_id		
vec_copy	Sec. 6.2.59.1 Vehicles CSCI SDD	
LOOKUP_POSITION	Sec. 2.9.1.1 See Appendix A	
LOOKUP_VELOCITY	Sec. 2.9.1.1 See Appendix A	

Table 2.6-76: collision_vehicle_rammed CSU [8.6.3.1.12]

2.6.3.1.13 detect_collision_tick CSU

This CSU checks to see if we hit anybody this tick and checks whether anybody hit us since the last tick.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*collision	pointer to COLLISION VARS	Sec. 2.9.1.2
*saf_vehicle	pointer to SAF VEHICLE VARS	Sec. 2.9.1.2
*entity	pointer to ENTITY VARS	Sec. 2.9.1.2

ReturnValues		
Return Value	Type	Meaning
TRUE	int	Collision has occurred.
FALSE	int	Collision has not occurred.
Calls		
Function	Where Described	
sign		
point in building	Sec. 2.12.1.17.2	
DEBUG COLLISION	Sec. 2.5.2.2 See Appendix A	
OBJ VEHICLEID	Sec. 2.9.1.1 See Appendix A	
OBJ POSITION	Sec. 2.9.1.1 See Appendix A	
FOR VEHICLES_WITHIN_N_GRIDS DO	Sec. 2.9.3.2 See Appendix A	
FARTHER3	Sec. 2.14.3.9 See Appendix A	
simnet_send collision		
vec copy	Sec. 2.6.2.59.1 Vehicles CSCI SDD	
LOOKUP POSITION	Sec. 2.9.1.1 See Appendix A	
LOOKUP_VELOCITY	Sec. 2.9.1.1 See Appendix A	

Table 2.6-77: detect_collision_tick CSU [8.6.3.1.13]

2.6.3.1.14 start_disengaging CSU

This CSU checks that backing up is appropriate and then does so.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*collision	pointer to COLLISION_VARS	Sec. 2.9.1.2
*saf_vehicle	pointer to SAF_VEHICLE_VARS	Sec. 2.9.1.2
*entity	pointer to ENTITY_VARS	Sec. 2.9.1.2
ReturnValues		
Return Value	Type	Meaning
TRUE	int	Backing up is appropriate.
FALSE	int	Backing up is not appropriate.
Calls		
Function	Where Described	
vec_init	Sec. 2.6.2.61.1 Vehicles CSCI SDD	
vec_sub	Sec. 2.6.2.65.1 Vehicles CSCI SDD	
interior_angle_between_vectors	Sec. 2.14.3.5.8	
VEC_SMALLER2	Sec. 2.14.3.9 See Appendix A	
saf_vehicle_sudden_stop	Sec. 2.6.1.1.27	
vec_add	Sec. 2.6.2.57.1 Vehicles CSCI SDD	
vec_normalize	Sec. 2.6.2.63.1 Vehicles CSCI SDD	
vec_copy	Sec. 2.6.2.59.1 Vehicles CSCI SDD	

Table 2.6-78: start_disengaging CSU [8.6.3.1.14]

2.6.3.1.15 phase_two_collision_check CSU

This CSU compares the position and velocity vectors of two vehicles and determines if evasive action is called for.

Parameters		
Parameters	Type	Where Typedef Declared
*my_pos	pointer to REAL	sim_types.h
*his_pos	pointer to REAL	sim_types.h
*my_vel	pointer to REAL	sim_types.h
*his_vel	pointer to REAL	sim_types.h
*secs_until_collision	pointer to REAL	sim_types.h
ReturnValues		
Return Value	Type	Meaning
TRUE	int	Evasive action is required.
FALSE	int	Evasive action not called for.
Calls		
Function	Where Described	
square	sim_macros.h	

Table 2.6-79: phase_two_collision_check CSU [8.6.3.1.15]

2.6.3.1.16 choose_skirt_point CSU

This CSU finds a way around an obstacle. If *chosen_dir* is 0, it chooses the best skirt point; if -1, it chooses the skirt point on the left; if 1, the skirt point on the right. *chosen_dir* is set to the direction decided upon so it can be used on the next pass, if one is necessary.

Parameters		
Parameters	Type	Where Typedef Declared
*pos	pointer to REAL	sim_types.h
*dir	pointer to REAL	sim_types.h
*obstacle	pointer to register RECT	Sec. 2.14.3.9
*skirt_point	pointer to register REAL	sim_types.h
*chosen_dir	pointer to int	Standard
Calls		
Function	Where Described	
DEBUG COLLISION	Sec. 2.5.2.2 See Appendix A	
which_side	Sec. 2.14.3.5.9	
interior_angle_between_vectors	Sec. 2.14.3.5.8	

Table 2.6-80: choose_skirt_point CSU [8.6.3.1.16]

2.6.3.2 driver.c CSC

/simnet/src/host/driver.c

The driver code handles the moving of the vehicle. It handles where the vehicle needs to go (but does NOT determine where that is), much like the pilot code for flying vehicles.

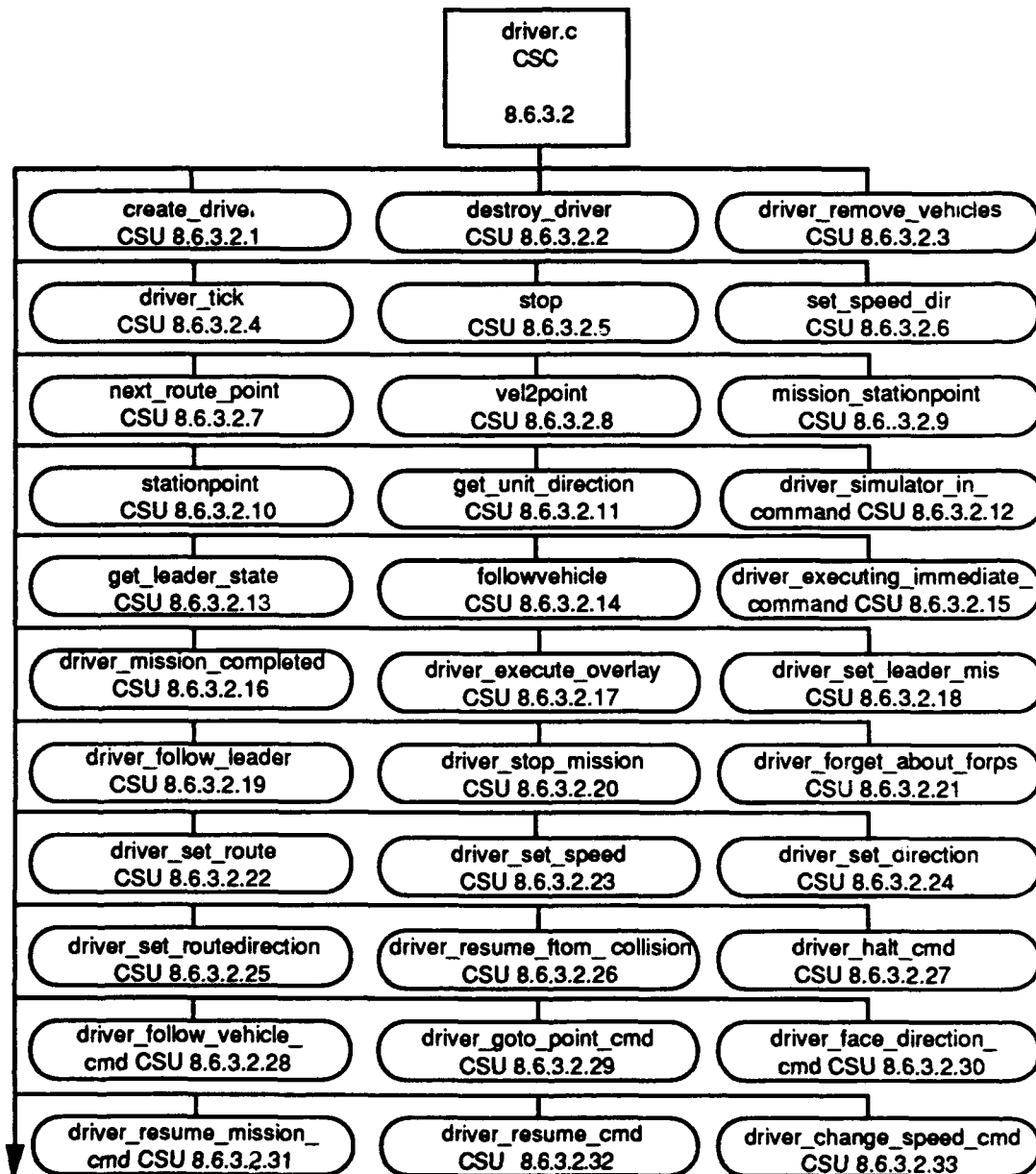


Figure 2.6-7: driver.c CSC Structure Part 1 of 2

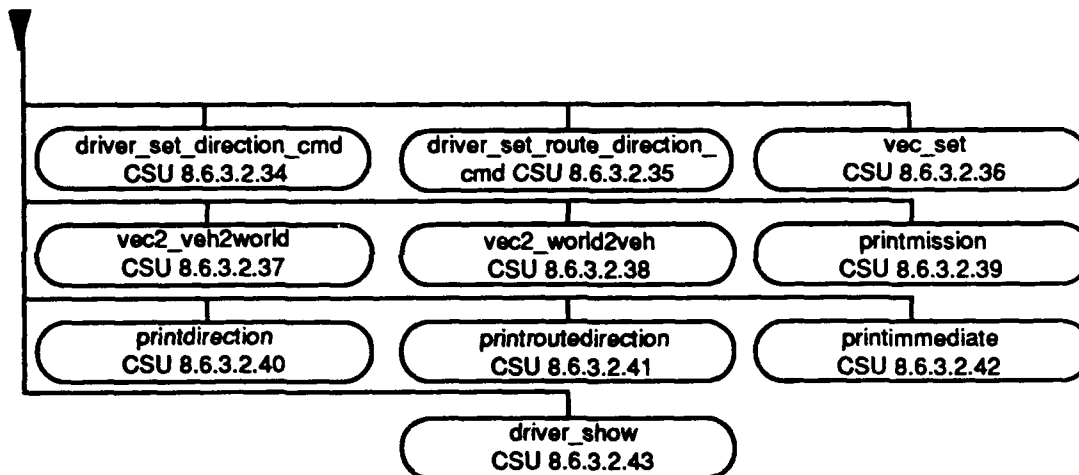


Figure 2.6-8: driver.c CSC Structure Part 2 of 2

In addition to the CSUs contained in this CSC there are three constants defined.

Constant	Value
LEADER CORRIDOR	5.0
CATCHUP TIME NORMAL	10.0
CATCHUP TIME BRIDGE	1.0

Table 2.6-81: driver.c Constants

2.6.3.2.1 create_driver CSU

This CSU creates a driver.

Parameters		
Parameters	Type	Where Typedef Declared
max_err2	REAL	sim_types.h
ReturnValues		
Return Value	Type	Meaning
driver	pointer to DRIVER_VARS	Created driver.
Calls		
Function	Where Described	
allocate_driver	Sec. 2.9.1.2 See Appendix A	
vec_set	Sec. 2.6.3.2.36	
vec_init	Sec. 2.6.2.61.1 Vehicles CSCI SDD	

Table 2.6-82: create_driver CSU [8.6.3.2.1]

2.6.3.2.2 destroy_driver CSU

This CSU destroys a driver by calling deallocate_driver.

Parameters		
Parameters	Type	Where Typedef Declared
*driver	pointer to DRIVER_VARS	
Calls		
Function	Where Described	
deallocate_driver	Sec. 2.9.1.2 See Appendix A	

Table 2.6-83: destroy_driver CSU [8.6.3.2.2]

2.6.3.2.3 driver_remove_vehicles CSU

This CSU removes the lead vehicle.

Parameters		
Parameters	Type	Where Typedef Declared
*driver	pointer to DRIVER_VARS	Sec. 2.9.1.2
num	int	Standard
v_list[]	unsigned int	Standard
Calls		
Function	Where Described	
LOOKUP_SAFOBJ	Sec. 2.9.1.1 See Appendix A	

Table 2.6-84: driver_remove_vehicles CSU [8.6.3.2.3]

2.6.3.2.4 driver_tick CSU

This CSU is the driver state machine.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
Errors		
Error Name	Reason for Error	
"Veh ... has lost his immediate leader"	There is no immediate leader for the vehicle.	
"driver ... in illegal immediate state"	Value in driver->immediate is none of the expected states.	
"Veh ... has lost its routepoint"	There is no routepoint for this driver.	
"Veh ... has no leader to follow"	There is no leadveh for this driver.	

Calls	
Function	Where Described
stop	Sec. 2.6.3.2.5
within_delta	Sec. 2.14.1.2.10
vel2point	Sec. 2.6.3.2.8
followvehicle	Sec. 2.6.3.2.14
ERROR_OUT	Sec. 2.5.2.2 See Appendix A

Table 2.6-85: driver_tick CSU [8.6.3.2.4]

2.6.3.2.5 stop CSU

This CSU causes a vehicle to stop.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*direction	pointer to REAL	sim_types.h
Calls		
Function	Where Described	
vec_copy	Sec. 2.6.2.59.1 Vehicles CSCI SDD	

Table 2.6-86: stop CSU [8.6.3.2.5]

2.6.3.2.6 set_speed_dir CSU

This CSU sets the speed and direction.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
speed	REAL	sim_types.h
direction	REAL	sim_types.h
Calls		
Function	Where Described	
vec_copy	Sec. 2.6.2.59.1 Vehicles CSCI SDD	
vec2_norm	Sec. 2.14.3.5.31	

Table 2.6-87: set_speed_dir CSU [8.6.3.2.6]

2.6.3.2.7 next_route_point CSU

This CSU sends the driver to the next route point.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
stop	Sec. 2.6.3.2.5	

Table 2.6-88: next_route_point CSU [8.6.3.2.7]

2.6.3.2.8 vel2point CSU

This CSU determines the speed required to get to the next point and sets it.

Parameters		
Parameters	Type	Where Typedef Declared
*point	pointer to REAL	sim_types.h
*position	pointer to REAL	sim_types.h
speed	REAL	sim_types.h
direction	int	Standard
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
vec2_sub	Sec. 2.14.3.5.23	
vec2_mag	Sec. 2.14.3.5.26	
min	Sec. 2.13.3.5 See Appendix A	
vec set	Sec. 2.6.3.2.36	
vec copy	Sec. 2.6.2.59.1 Vehicles CSCI SDD	

Table 2.6-89: vel2point CSU [8.6.3.2.8]

2.6.3.2.9 mission_stationpoint CSU

Parameters		
Parameters	Type	Where Typedef Declared
*driver	pointer to DRIVER_VARS	Sec. 2.9.1.2
point	VECTOR	sim_types.h
ReturnValues		
Return Value	Type	Meaning
TRUE	int	Success.
FALSE	int	Not the lead vehicle.

Calls	
Function	Where Described
stationpoint	Sec. 2.6.3.2.10

Table 2.6-90: mission_stationpoint CSU [8.6.3.2.9]

2.6.3.2.10 stationpoint CSU

Parameters		
Parameters	Type	Where Typedef Declared
leadpos	VECTOR	sim_types.h
leadvel	VECTOR	sim_types.h
leaddir	VECTOR	sim_types.h
offsetv	VECTOR	sim_types.h
point	VECTOR	sim_types.h
Calls		
Function	Where Described	
vec2_copy	Sec. 2.14.3.5.29	
vec_normalize	Sec. 2.6.2.63.1 Vehicles CSCI SDD	
vec2_veh2world	Sec. 2.6.3.2.37	
vec2_add	Sec. 2.14.3.5.22	

Table 2.6-91: stationpoint CSU [8.6.3.2.10]

2.6.3.2.11 get_unit_direction CSU

This CSU determines the direction of the lead vehicle.

Parameters		
Parameters	Type	Where Typedef Declared
*leadveh	pointer to SAF_OBJECT	Sec. 2.9.1.1
unitdir	VECTOR	sim_types.h
*unitspeed	pointer to REAL	sim_types.h
samity	int	Standard
Calls		
Function	Where Described	
get_unit_direction	Sec. 2.6.3.2.11	
vec_copy	Sec. 2.6.2.59.1 Vehicles CSCI SDD	
vec2_mag	Sec. 2.14.3.5.26	
vec2_dot	Sec. 2.14.3.5.24	

Table 2.6-92: get_unit_direction CSU [8.6.3.2.11]

2.6.3.2.12 driver_simulator_in_command CSU

Parameters		
Parameters	Type	Where Typedef Declared
*driver	pointer to DRIVER_VARS	Sec. 2.9.1.2
sanity	int	Standard
ReturnValues		
Return Value	Type	Meaning
TRUE	int	
FALSE	int	
leadveh->driver && driver_simulator_in_command(...)	int	Recursion.
Calls		
Function	Where Described	
driver_simulator_in_command	Sec. 2.6.3.2.12	

Table 2.6-93: driver_simulator_in_command CSU [8.6.3.2.12]**2.6.3.2.13 get_leader_state CSU**

This CSU determines the state of the lead vehicle.

Parameters		
Parameters	Type	Where Typedef Declared
*leadveh	pointer to SAF_OBJECT	Sec. 2.9.1.1
pos	VECTOR	sim_types.h
vel	VECTOR	sim_types.h
dir	VECTOR	sim_types.h
*speed	REAL	sim_types.h
**forp	pointer to pointer to ROUTEPOINT	Sec. 2.10.2.5
*leader_route_dir	pointer to int	Standard
ReturnValues		
Return Value	Type	Meaning
get_leader_state(...)	unsigned short	Recursion.
leader_behavior	unsigned short	Resulting leader behavior.
Calls		
Function	Where Described	
vec2_copy	Sec. 2.14.3.5.29	
vec2_mag	Sec. 2.14.3.5.26	
vec2_dot	Sec. 2.14.3.5.24	
LOOKUP_VEHICLE	Sec. 2.9.3.2 See Appendix A	
get_leader_state	Sec. 2.6.3.2.13	
remote_next_road_point	Sec. 2.7.1.9	
vec_normalize	Sec. 2.6.2.63.1 Vehicles CSCI SDD	

Table 2.6-94: get_leader_state CSU [8.6.3.2.13]

2.6.3.2.14 followvehicle CSU

This CSU causes the vehicles to follow the lead vehicle.

Parameters		
Parameters	Type	Where Typedef Declared
position	VECTOR	sim_types.h
*leadveh	pointer to SAF OBJECT	Sec. 2.9.1.1
offset	VECTOR	sim_types.h
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
**forp	pointer to pointer to ROUTEPOINT	Sec. 2.10.2.5
forp_dir	pointer to int	Standard
Calls		
Function	Where Described	
saf_vehicle_doing_collision_stuff	Sec. 2.6.1.1.43	
set speed dir	Sec. 2.6.3.2.6	
get leader state	Sec. 2.6.3.2.13	
within delta	Sec. 2.14.1.2.10	
abs	Sec. 2.6.7.3 & Sec. 2.13.3.2 See Appendix A	
vec2_sub	Sec. 2.14.3.5.23	
vec2_dot	Sec. 2.14.3.5.24	
set speed dir	Sec. 2.6.3.2.6	
square	sim_macros.h	
almost_eq	Sec. 2.14.3.9 See Appendix A	
min	Sec. 2.13.3.5 See Appendix A	
max	Sec. 2.13.3.5 See Appendix A	
vel2point	Sec. 2.6.3.2.8	
get unit direction	Sec. 2.6.3.2.11	
vec2_scale	Sec. 2.14.3.5.28	
vec_copy	Sec. 2.6.2.59.1 Vehicles CSCI SDD	
vec2_norm	Sec. 2.14.3.5.31	
vec2_world2veh	Sec. 2.6.3.2.38	
vec2_add	Sec. 2.14.3.5.22	
stop	Sec. 2.6.3.2.5	
vec_scale	Sec. 2.6.2.63.1 Vehicles CSCI SDD	

Table 2.6-95: followvehicle CSU [8.6.3.2.14]

2.6.3.2.15 driver_executing_immediate_command CSU

Parameters		
Parameters	Type	Where Typedef Declared
*driver	pointer to DRIVER_VARS	Sec. 2.9.1.2

ReturnValues		
Return Value	Type	Meaning
driver->immediate != IDLE	int	Driver executing immediate command.

Table 2.6-96: driver_executing_immediate_command CSU [8.6.3.2.15]

2.6.3.2.16 driver_mission_completed CSU

This CSU returns TRUE if the driver is immediately inactive, the driver is the driver mission is idle, lost in any way, or if the end of route has been reached. It returns FALSE otherwise.

Parameters		
Parameters	Type	Where Typedef Declared
*driver	pointer to DRIVER_VARS	Sec. 2.9.1.2
ReturnValues		
Return Value	Type	Meaning
TRUE	int	Driver is immediately inactive, or driver mission is idle, end of route, or lost in any way.
FALSE	int	Driver is anything else.

Table 2.6-97: driver_mission_completed CSU [8.6.3.2.16]

2.6.3.2.17 driver_execute_overlay CSU

This CSU causes the driver to execute the current overlay.

Parameters		
Parameters	Type	Where Typedef Declared
*unit	pointer to SAF_OBJECT	Sec. 2.9.1.1
*overlay	pointer to OVERLAY	Sec. 2.10.2.2
*cm	pointer to CONTROL MEASURE	Sec. 2.10.2.2
Errors		
Error Name	Reason for Error	
"Cannot execute anything but route"	Control measure is not of type route.	
Calls		
Function	Where Described	
ERROR_OUT	Sec. 2.5.2.2 See Appendix A	
driver resume mission cmd	Sec. 2.6.3.2.31	

Table 2.6-98: driver_execute_overlay CSU [8.6.3.2.17]

2.6.3.2.18 driver_set_leader_mis CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer toSAF_OBJECT	Sec. 2.9.1.1
vid	unsigned int	Standard
x_offset	int	Standard
y_offset	int	Standard
Errors		
Error Name	Reason for Error	
"Veh ... told to follow nonexistant vehicle"	No lead vehicle.	
"Veh ... cannot follow itself"	This vehicle is the lead vehicle.	
Calls		
Function	Where Described	
LOOKUP_SAFOBJ	Sec. 2.9.1.1 See Appendix A	
ERROR_OUT	Sec. 2.5.2.2 See Appendix A	
abort		

Table 2.6-99: driver_set_leader_mis CSU [8.6.3.2.18]

2.6.3.2.19 driver_follow_leader CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer toSAF_OBJECT	Sec. 2.9.1.1
Errors		
Error Name	Reason for Error	
"Veh ... Unable to follow, no leader"	No lead vehicle.	
Calls		
Function	Where Described	
ERROR_OUT	Sec. 2.5.2.2 See Appendix A	
abort		

Table 2.6-100: driver_follow_leader CSU [8.6.3.2.19]

2.6.3.2.20 driver_stop_mission CSU

Parameters		
Parameters	Type	Where Typedef Declared
*driver	pointer to DRIVER_VARS	Sec. 2.9.1.2

Table 2.6-101: driver_stop_mission CSU [8.6.3.2.20]

2.6.3.2.21 driver_forget_about_forps CSU

Parameters		
Parameters	Type	Where Typedef Declared
*driver	pointer to DRIVER_VARS	Sec. 2.9.1.2

Table 2.6-102: driver_forget_about_forps CSU [8.6.3.2.21]**2.6.3.2.22 driver_set_route CSU**

This CSU sets the route.

Parameters		
Parameters	Type	Where Typedef Declared
*driver	pointer to DRIVER_VARS	Sec. 2.9.1.2
*route	pointer to ROUTE	Sec. 2.10.2.5
*routept	pointer to ROUTEPOINT	Sec. 2.10.2.5

Table 2.6-103: driver_set_route CSU [8.6.3.2.22]**2.6.3.2.23 driver_set_speed CSU**

This CSU sets the speed to speed.

Parameters		
Parameters	Type	Where Typedef Declared
*driver	pointer to DRIVER_VARS	Sec. 2.9.1.2
speed	REAL	sim_types.h
ReturnValues		
Return Value	Type	Meaning
ldriver->useimspeed	int	new speed

Table 2.6-104: driver_set_speed CSU [8.6.3.2.23]**2.6.3.2.24 driver_set_direction CSU**

This CSU sets the direction to direction.

Parameters		
Parameters	Type	Where Typedef Declared
*driver	pointer to DRIVER_VARS	Sec. 2.9.1.2
direction	int	Standard

Table 2.6-105: driver_set_direction CSU [8.6.3.2.24]

2.6.3.2.25 driver_set_routedirection CSU

This CSU sets the routedirection to routedirection.

Parameters		
Parameters	Type	Where Typedef Declared
*driver	pointer to DRIVER_VARS	Sec. 2.9.1.2
routedirection	int	Standard

Table 2.6-106: driver_set_routedirection CSU [8.6.3.2.25]

2.6.3.2.26 driver_resume_from_collision CSU

Parameters		
Parameters	Type	Where Typedef Declared
*driver	pointer to DRIVER_VARS	Sec. 2.9.1.2
*box	pointer to RECT	Sec. 2.14.3.9
Calls		
Function	Where Described	
point_in_rect	Sec. 2.14.3.9 See Appendix A	

Table 2.6-107: driver_resume_from_collision CSU [8.6.3.2.26]

2.6.3.2.27 driver_halt_cmd CSU

This CSU causes the driver to stop.

Parameters		
Parameters	Type	Where Typedef Declared
*driver	pointer to DRIVER_VARS	Sec. 2.9.1.2

Table 2.6-108: driver_halt_cmd CSU [8.6.3.2.27]

2.6.3.2.28 driver_follow_vehicle_cmd CSU

This CSU causes the driver to follow a lead vehicle.

Parameters		
Parameters	Type	Where Typedef Declared
*driver	pointer to DRIVER_VARS	Sec. 2.9.1.2
vid	unsigned int	Standard
xoff	int	Standard
yoff	int	Standard
Errors		
Error Name	Reason for Error	
"Veh ... told to follow nonexistent vehicle"	No lead vehicle.	

Calls	
Function	Where Described
LOOKUP SAFOBJ	Sec. 2.9.1.1 See Appendix A
ERROR_OUT	Sec. 2.5.2.2 See Appendix A
abort	
vec set	Sec. 2.6.3.2.36

Table 2.6-109: driver_follow_vehicle_cmd CSU [8.6.3.2.28]

2.6.3.2.29 driver_goto_point_cmd CSU

This CSU causes the driver to go to a designated point.

Parameters		
Parameters	Type	Where Typedef Declared
*driver	pointer to DRIVER_VARS	Sec. 2.9.1.2
x	REAL	sim_types.h
y	REAL	sim_types.h
backwardp	int	Standard
Calls		
Function	Where Described	
vec set	Sec. 2.6.3.2.36	

Table 2.6-110: driver_goto_point_cmd CSU [8.6.3.2.29]

2.6.3.2.30 driver_face_direction_cmd CSU

This CSU causes the driver to face a designated direction.

Parameters		
Parameters	Type	Where Typedef Declared
*driver	pointer to DRIVER_VARS	Sec. 2.9.1.2
mathradians	REAL	sim_types.h
Calls		
Function	Where Described	
vec set	Sec. 2.6.3.2.36	

Table 2.6-111: driver_face_direction_cmd CSU [8.6.3.2.30]

2.6.3.2.31 driver_resume_mission_cmd CSU

This CSU causes the driver to resume the mission.

Parameters		
Parameters	Type	Where Typedef Declared
*driver	pointer to DRIVER_VARS	Sec. 2.9.1.2

Table 2.6-112: driver_resume_mission_cmd CSU [8.6.3.2.31]

2.6.3.2.32 driver_resume_cmd CSU

This CSU is obsolete.

Parameters		
Parameters	Type	Where Typedef Declared
*driver	pointer to DRIVER_VARS	Sec. 2.9.1.2
Calls		
Function	Where Described	
driver_resume_mission_cmd	Sec. 2.6.3.2.31	

Table 2.6-113: driver_resume_cmd CSU [8.6.3.2.32]

2.6.3.2.33 driver_change_speed_cmd CSU

This CSU causes the driver to change speed.

Parameters		
Parameters	Type	Where Typedef Declared
*driver	pointer to DRIVER_VARS	Sec. 2.9.1.2
speed	REAL	sim_types.h

Table 2.6-114: driver_change_speed_cmd CSU [8.6.3.2.33]

2.6.3.2.34 driver_set_direction_cmd CSU

This CSU sets the new direction.

Parameters		
Parameters	Type	Where Typedef Declared
*driver	pointer to DRIVER_VARS	Sec. 2.9.1.2
direction	int	Standard
Calls		
Function	Where Described	
driver_set_direction	Sec. 2.6.3.2.24	

Table 2.6-115: driver_set_direction_cmd CSU [8.6.3.2.34]

2.6.3.2.35 driver_set_route_direction_cmd CSU

This CSU sets the new route direction.

Parameters		
Parameters	Type	Where Typedef Declared
*driver	pointer to DRIVER_VARS	Sec. 2.9.1.2
routedirection	int	Standard
Calls		
Function	Where Described	
driver_set_routedirection	Sec. 2.6.3.2.25	

Table 2.6-116: driver_set_route_direction_cmd CSU [8.6.3.2.35]

2.6.3.2.36 vec_set CSU

This CSU assigns values to the components of a vector.

Parameters		
Parameters	Type	Where Typedef Declared
v	VECTOR	sim_types.h
x	REAL	sim_types.h
y	REAL	sim_types.h
z	REAL	sim_types.h

Table 2.6-117: vec_set CSU [8.6.3.2.36]

2.6.3.2.37 vec2_vch2world CSU

Parameters		
Parameters	Type	Where Typedef Declared
directionx	REAL	sim_types.h
directiony	REAL	sim_types.h
offsetv	VECTOR	sim_types.h
offsetw	VECTOR	sim_types.h
Calls		
Function	Where Described	
vec_copy	Sec. 2.6.2.59.1 Vehicles CSCI SDD	

Table 2.6-118: vec2_vch2world CSU [8.6.3.2.37]

2.6.3.2.38 vec2_world2veh CSU

Parameters		
Parameters	Type	Where Typedef Declared
directionx	REAL	sim_types.h
directiony	REAL	sim_types.h
offsetv	VECTOR	sim_types.h
offsetw	VECTOR	sim_types.h
Calls		
Function	Where Described	
vec_copy	Sec. 2.6.2.59.1 Vehicles CSCI SDD	

Table 2.6-119: vec2_world2veh CSU [8.6.3.2.38]**2.6.3.2.39 printmission CSU**

This CSU prints the current status of a mission.

Parameters		
Parameters	Type	Where Typedef Declared
mode	int	Standard

Table 2.6-120: printmission CSU [8.6.3.2.39]**2.6.3.2.40 printdirection CSU**

This CSU prints the current vehicle direction.

Parameters		
Parameters	Type	Where Typedef Declared
mode	int	Standard

Table 2.6-121: printdirection CSU [8.6.3.2.40]**2.6.3.2.41 printroutedirection CSU**

This CSU prints the current route direction.

Parameters		
Parameters	Type	Where Typedef Declared
mode	int	Standard

Table 2.6-122: printroutedirection CSU [8.6.3.2.41]

2.6.3.2.42 printimmediate CSU

This CSU prints the current state of a saf object.

Parameters		
Parameters	Type	Where Typedef Declared
mode	int	Standard

Table 2.6-123: printimmediate CSU [8.6.3.2.42]

2.6.3.2.43 driver_show CSU

This CSU prints the current state of the driver.

Parameters		
Parameters	Type	Where Typedef Declared
*driver	pointer to DRIVER_VARS	Sec. 2.9.1.2
flags	int	Standard
Calls		
Function	Where Described	
printimmediate	Sec. 2.6.3.2.42	
OBJ_VEHICLEID	Sec. 2.9.1.1 See Appendix A	
print_vector	Sec. 2.14.3.5.2	
print_routepoint	Sec. 2.10.2.4.16	
printmission	Sec. 2.6.3.2.39	
printdirection	Sec. 2.6.3.2.40	
printroutedirection	Sec. 2.6.3.2.41	
print_route	Sec. 2.10.2.4.17	

Table 2.6-124: driver_show CSU [8.6.3.2.43]

2.6.3.3 driver.h CSU

/simnet/src/host/driver.h

This file contains all the symbolic constants used by the driver.c CSU in the following table.

Constant	Value
IDLE	0 /* follow mode */
ENDROUTE	1
LOSTROUTE	2
LOSTROUTEPOINT	3
LOSTLEADER	6
FOLLOWROUTE	8
FOLLOWVEHICLE	9
FORWARD	0 /* direction, routedirection */
REVERSE	1
INACTIVE	0 /* immediate */
GOTOPOINT	1
ATPOINT	2
FACEDIRECTION	3
HALT	4
IMFOLLOWVEH	5
IMLOSTVEH	6
LEADERS ACTUAL	0 /* follow algorithms */
LEADERS DESIRED	1
LEADERS LEADER ACTUAL	2
LEADER ON ROAD	1
LEADER ON DIRT	2
LEADER ON BRIDGE	3
GRND MIN TURN SPEED	2.0 /* general */
GRND MAX BACKUP DIST2	400.0
GRND MAX BACKUP SPEED	2.0

Table 2.6-125: driver.h Constants

2.6.3.4 groundveh.c CSC

/simnet/src/host/groundveh.c

This file contains all of the CSUs specific to ground vehicles, including creation, deletion, and handling everything being done by a ground vehicle during each tick.

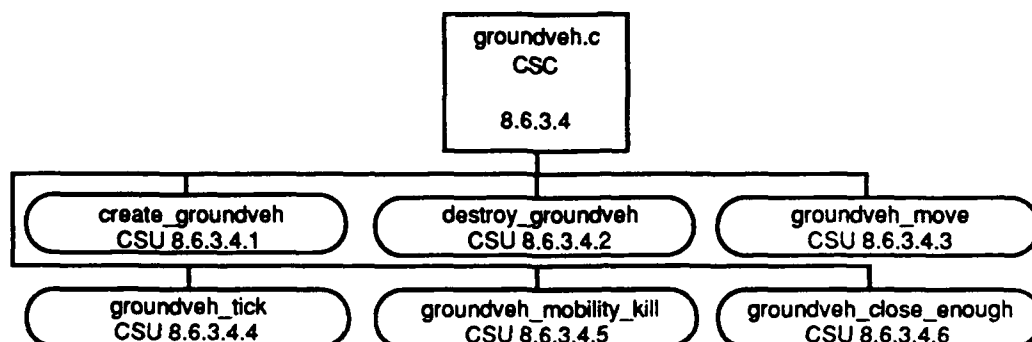


Figure 2.6-9: groundveh.c.c CSC Structure

The sine of 45 degrees is defined as a constant within the groundveh_tick CSU (#define SIN_45 0.707).

2.6.3.4.1 create_groundveh CSU

This CSU creates a ground vehicle.

Parameters		
Parameters	Type	Where Typedef Declared
*table	pointer to DATA_UNION	Sec. 2.1.1.5
*saf_vehicle	pointer to SAF_VEHICLE_VARS	Sec. 2.9.1.2
ReturnValues		
Return Value	Type	Meaning
groundveh	pointer to GROUNDVEH_VARS	Created ground vehicle.
Calls		
Function	Where Described	
allocate_groundveh	Sec. 2.9.1.2 See Appendix A	
mat_rot_init		
deg_to_rad	sim_macros.h	
mat_mat_mul		
kph_to_mps	Sec. 2.13.3.1 See Appendix A	
ft_float	Sec. 2.14.1.2.12	

Table 2.6-126: create_groundveh CSU [8.6.3.4.1]

2.6.3.4.2 destroy_groundveh CSU

This CSU deallocates the memory allocated for the ground vehicle passed as the parameter.

Parameters		
Parameters	Type	Where Typedef Declared
*groundveh	pointer to GROUNDVEH_VARS	Sec. 2.9.1.2
Calls		
Function	Where Described	
deallocate_groundveh	Sec. 2.9.1.2 See Appendix A	

Table 2.6-127: destroy_groundveh CSU [8.6.3.4.2]

2.6.3.4.3 groundveh_move CSU

This CSU coordinates the movements of a ground vehicle.

Parameters		
Parameters	Type	Where Typedef Declared
*saf_vehicle	pointer to SAF_VEHICLE_VARS	Sec. 2.9.1.2
*groundveh	pointer to GROUNDVEH_VARS	Sec. 2.9.1.2
*vehicle	pointer to VEHICLE_VARS	Sec. 2.9.1.2
*entity	pointer to ENTITY_VARS	Sec. 2.9.1.2
time_step	REAL	sim_types.h
on_road	int	Standard
Calls		
Function	Where Described	
angle_between_vectors	Sec. 2.14.3.5.7	
abs	Sec. 2.6.7.3 & Sec. 2.13.3.2 See Appendix A	
min	Sec. 2.13.3.5 See Appendix A	
vec_z_rotate		
sign		
RANGE_CLIP	Sec. 2.14.3.9 See Appendix A	
almost_eq	Sec. 2.14.3.9 See Appendix A	
s_atan2	Sec. 2.14.3.9 See Appendix A	
tdb_place_vehicle	Sec. 2.21.7.20.5	
report_error_from_tdb_once	Sec. 2.14.1.2.4	
mat_rot_init	Sec. 2.6.2.47.1 Vehicles CSCI SDD	
get_soil_type	Sec. 2.14.1.2.1	
vec_scale	Sec. 2.6.2.64.1 Vehicles CSCI SDD	
copy_matrix_row_to_vector	Sec. 2.14.3.5.18	

Table 2.6-128: groundveh_move CSU [8.6.3.4.3]

2.6.3.4.4 groundveh_tick CSU

This CSU models a ground vehicle on a tick by tick basis.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
Errors		
Error Name	Reason for Error	
"Unknown soil type	Value ingroundveh->last_soil_type is none of the expected soil types.	
Calls		
Function	Where Described	
groundveh_move	Sec. 2.6.3.4.3	
mat mat mul	Sec. 2.6.2.32.1 Vehicles CSCI SDD	
ERROR_OUT	Sec. 2.5.2.2 See Appendix A	
composite_note_leader_state	Sec. 2.8.1.3.36	
composite_inferior_changed_status	Sec. 2.8.1.3.37	
abs	Sec. 2.6.7.3 & Sec. 2.13.3.2 See Appendix A	
sign		
RANGE_CLIP	Sec. 2.14.3.9 See Appendix A	
DEBUG_GROUND	Sec. 2.5.2.2 See Appendix A	
groundveh_move	Sec. 2.6.3.4.3	
water_check	Sec. 2.12.1.4.4	
vec_init	Sec. 2.6.2.61.1 Vehicles CSCI SDD	

Table 2.6-129: groundveh_tick CSU [8.6.3.4.4]

2.6.3.4.5 groundveh_mobility_kill CSU

This CSU renders a ground vehicle immobile.

Parameters		
Parameters	Type	Where Typedef Declared
*groundveh	pointer to GROUNDVEH_VARS	Sec.2 9.1.2

Table 2.6-130: groundveh_mobility_kill CSU [8.6.3.4.5]

2.6.3.4.6 groundveh_close_enough CSU

This CSU determines if the vehicle is close enough to a given point.

Parameters		
Parameters	Type	Where Typedef Declared
*saf_vehicle	pointer to SAF_VEHICLE_VARS	Sec. 2.9.1.2
*entity	pointer to ENTITY_VARS	Sec. 2.9.1.2
*despos	REAL	sim_types.h
ReturnValues		
Return Value	Type	Meaning
within_delta(...)	int	the vehicle was within the required tolerance.
Calls		
Function	Where Described	
within_delta	Sec. 2.14.1.2.10	

Table 2.6-131: groundveh_close_enough CSU [8.6.3.4.6]

2.6.4 Air Maneuver CSC

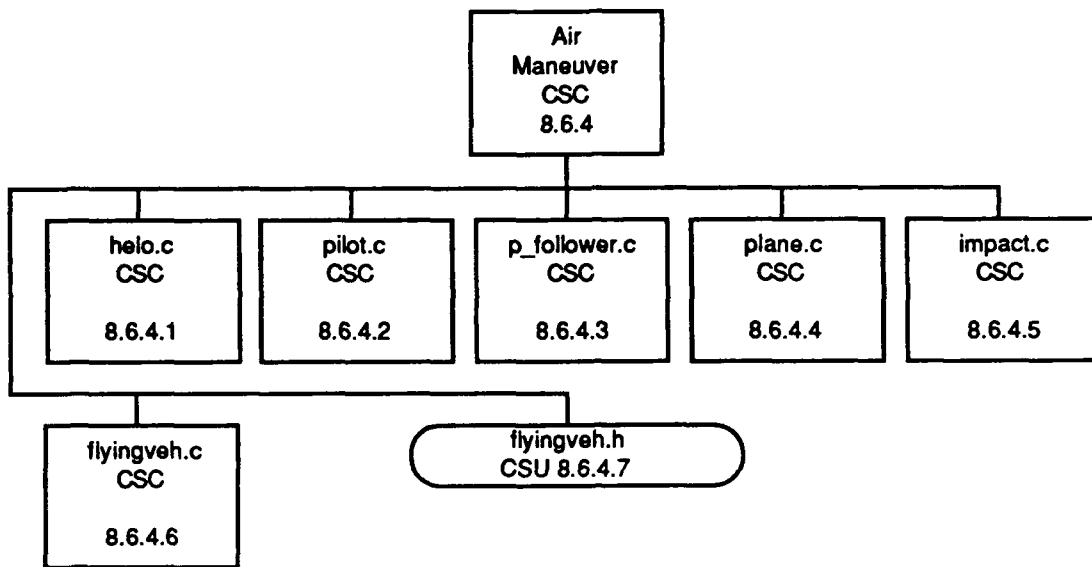


Figure 2.6-10: Air Maneuver CSC Structure

2.6.4.1 helo.c CSC

/simnet/src/host/helo.c

This CSC handles everything that needs to be done by a helicopter each tick. This consists of a single CSU `helo_tick()`, the constant definition for the maximum helicopter power, the values in three, three element arrays, and values for six REAL variables.

Constant	Value
MAX HELICOPTER POWER	500000 /** Newtons **/

Table 2.6-132: helo.c Maximum Helicopter Power

Array	Values
REAL H INERTIA[]	{ 10000.0, 5000.0, 10000.0 }
REAL H K1[]	{ 6000.0, 4000.0, 6000.0 }
REAL H K2[]	{ 18000.0, 12000.0, 13000.0 }

Table 2.6-133: helo.c Array Constants

Variable	Value
REAL H K4	.03
REAL H K5	.05
REAL H K6	.03
REAL H K7	2 /** Air drag **/
REAL H K8	40 /** Air drag **/
REAL H KP	20000.0

Table 2.6-134: helo.c Assigned Variables

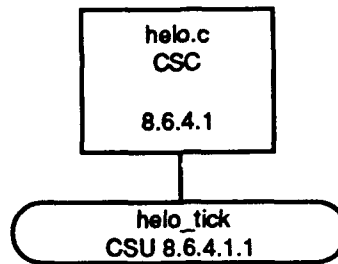


Figure 2.6-11: helo.c CSC Structure

2.6.4.1.1 helo_tick CSU

This CSU is responsible for the tick by tick simulation of the helicopters.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
min	Sec. 2.13.3.6 See Appendix A	
max	Sec. 2.13.3.6 See Appendix A	
abs	Sec. 2.6.7.3 & Sec. 2.13.3.2 See Appendix A	
almost_eq	Sec. 2.14.3.9 See Appendix A	
square	sim_macros.h	
sign		
s_atan2	Sec. 2.14.3.9 See Appendix A	
induce tail spin	Sec. 2.6.4.6.10	
hull_to_world_from_orientation	Sec. 2.6.4.6.8	
RANGE CLIP	Sec. 2.14.3.9 See Appendix A	
vec_mag3	sim_macros.j	
copy matrix row to vector	Sec. 2.14.3.5.18	
vec_init	Sec. 2.6.2.61.1 Vehicles CSCI SDD	

Table 2.6-135: helo_tick CSU [8.6.4.1.1]

2.6.4.2 pilot.c CSC

/simnet/src/host/pilot.c

The pilot code handles the moving of an air vehicle; where it needs to go (NOT determining where that is), much like the driver code for ground vehicles

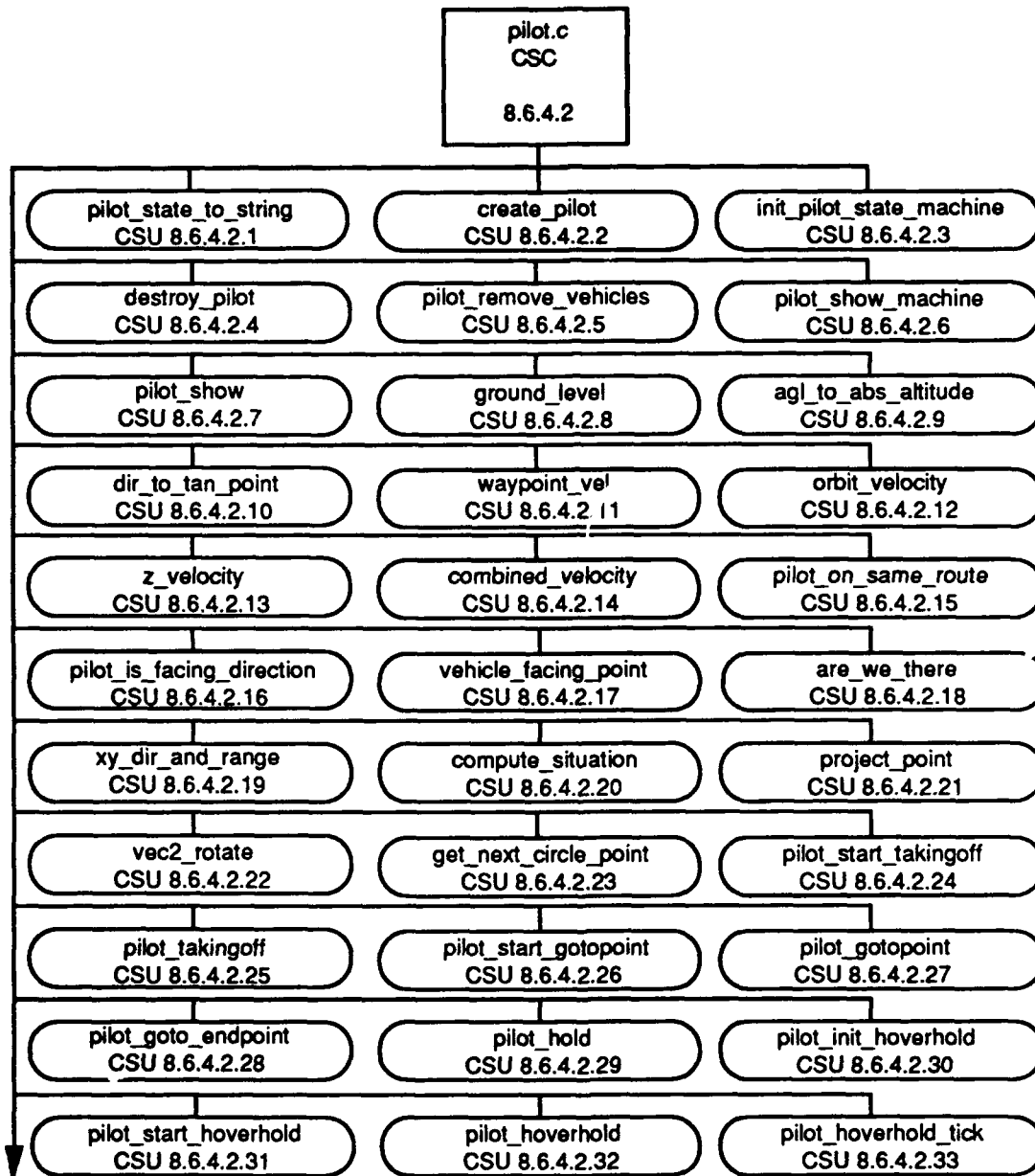


Figure 2.6-12: pilot.c CSC Structure Part 1 of 3

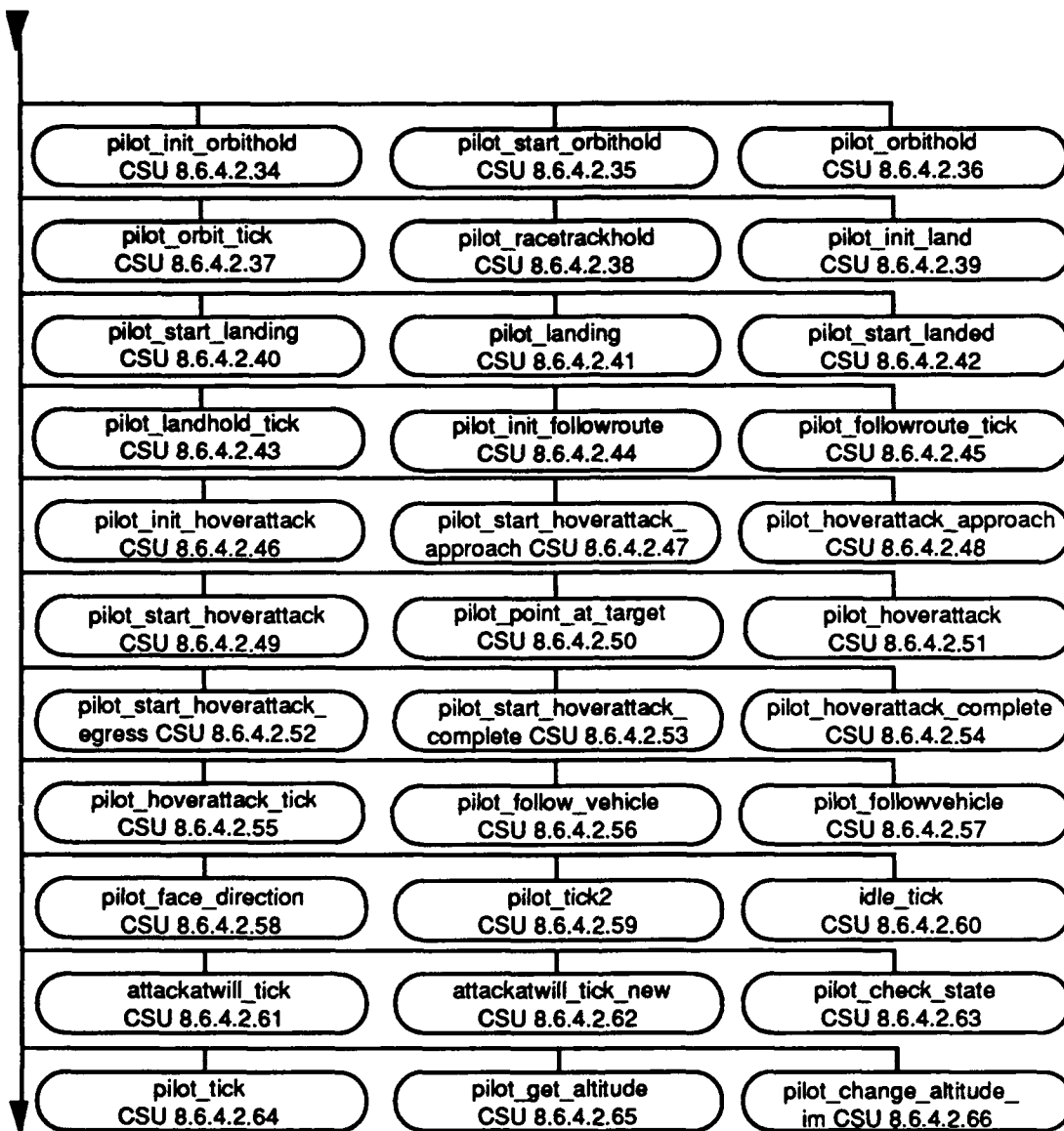


Figure 2.6-13: pilot.c CSC Structure Part 2 of 3

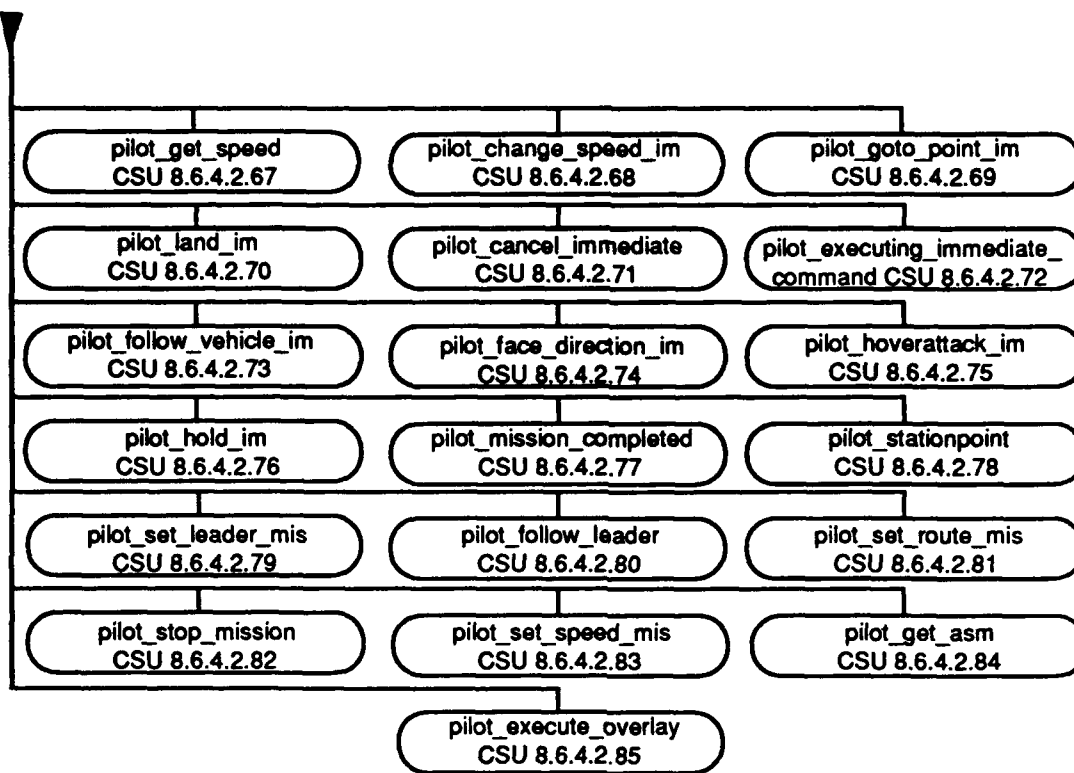


Figure 2.6-14: pilot.c CSC Structure part 3 of 3

2.6.4.2.1 pilot_state_to_string CSU

Parameters		
Parameters	Type	Where Typedef Declared
state	int	Standard
ReturnValues		
Return Value	Type	Meaning
"idle"	pointer to char	Pilot idle.
"hover hold"	pointer to char	Pilot holding in a hover.
"orbit hold"	pointer to char	Pilot holding in a circle.
"race track hold"	pointer to char	Pilot holding in a race track pattern.
"land hold"	pointer to char	Pilot holding by landing.
"follow route"	pointer to char	Pilot following a route.
"hover attack"	pointer to char	Pilot attacking while hovering.
"running attack"	pointer to char	Pilot attacking while moving.
"follow vehicle"	pointer to char	Pilot following vehicle.
"follow leader"	pointer to char	Pilot following leader.
"face direction"	pointer to char	Pilot turning.
"landed"	pointer to char	Pilot landed.
"taking off"	pointer to char	Pilot taking off.
"go to point"	pointer to char	Pilot going to a point.
"complete"	pointer to char	Pilot mission complete.
"landing"	pointer to char	Pilot landing.
"approach"	pointer to char	Pilot executing approach.
"egress"	pointer to char	Pilot breaking off hover attack.
"formation hover"	pointer to char	Pilot hovering in formation.
"unknown state"	pointer to char	Pilot state unknown.

Table 2.6-136: pilot_state_to_string CSU [8.6.4.2.1]

2.6.4.2.2 create_pilot CSU

This CSU creates a pilot.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*table	pointer to DATA_UNION	Sec. 2.1.1.5
type	int	Standard
ReturnValues		
Return Value	Type	Meaning
pilot	pointer to PILOT_VARS	Created pilot.

Calls	
Function	Where Described
allocate_pilot	Sec. 2.9.1.2 See Appendix A
init_pilot_state_machine	Sec. 2.6.4.2.3
vec_copy	Sec. 2.
vec_init	Sec. 2.

Table 2.6-137: create_pilot CSU [8.6.4.2.2]

2.6.4.2.3 init_pilot_state_machine CSU

This CSU initializes the pilot state machine.

Parameters		
Parameters	Type	Where Typedef Declared
*asm	pointer to AIR_SM	Sec. 2.6.4.7
type	int	Standard
Errors		
Error Name	Reason for Error	
ERROR_ABORT	Aircraft type is neither a helicopter nor a plane.	
Calls		
Function	Where Described	
vec_init	Sec. 2.6.2.61.1 Vehicles CSCI SDD	
vec_set	Sec. 2.6.3.2.36	

Table 2.6-138: init_pilot_state_machine CSU [8.6.4.2.3]

2.6.4.2.4 destroy_pilot CSU

Parameters		
Parameters	Type	Where Typedef Declared
*pilot	pointer to PILOT_VARS	
Calls		
Function	Where Described	
deallocate_pilot	Sec. 2.9.1.2 See Appendix A	

Table 2.6-139: destroy_pilot CSU [8.6.4.2.4]

2.6.4.2.5 pilot_remove_vehicles CSU

Parameters		
Parameters	Type	Where Typedef Declared
*pilot	pointer to PILOT_VARS	Sec. 2.9.1.2
num	int	Standard
v_list[]	unsigned int	Standard

Calls	
Function	Where Described
LOOKUP_SAFOBJ	Sec. 2.9.1.1 See Appendix A

Table 2.6-140: pilot_remove_vehicles CSU [8.6.4.2.5]

2.6.4.2.6 pilot_show_machine CSU

Parameters		
Parameters	Type	Where Typedef Declared
*asm	pointer to AIR_SM	Sec. 2.6.4.7
Calls		
Function	Where Described	
pilot_state_to_string	Sec. 2.6.4.2.1	
print_vector	Sec. 2.14.3.5.2	

Table 2.6-141: pilot_show_machine CSU [8.6.4.2.6]

2.6.4.2.7 pilot_show CSU

Parameters		
Parameters	Type	Where Typedef Declared
*pilot	pointer to PILOT_VARS	Sec. 2.9.1.2
flags	int	Standard
Calls		
Function	Where Described	
pilot_show_machine	Sec. 2.6.4.2.6	
print_vector	Sec. 2.14.3.5.2	

Table 2.6-142: pilot_show CSU [8.6.4.2.7]

2.6.4.2.8 ground_level CSU

This CSU determines the elevation above mean sea level at the specified *position*.

Parameters		
Parameters	Type	Where Typedef Declared
position	VECTOR	sim_types.h
extend	int	Standard
ReturnValues		
Return Value	Type	Meaning
vtemp[Z]	REAL	Elevation above msl.

Calls	
Function	Where Described
copy_xy_on_tdb	Sec. 2.14.1.2.5
vec2_copy	Sec. 2.14.3.5.29
tdb_get_gl	Sec. 2.14.1.2.2

Table 2.6-143: ground_level CSU [8.6.4.2.8]

2.6.4.2.9 agl_to_abs_altitude CSU

This CSU converts height above ground at the specified *position* to absolute altitude.

Parameters		
Parameters	Type	Where Typedef Declared
position	VECTOR	sim_types.h
agl	REAL	sim_types.h
ReturnValues		
Return Value	Type	Meaning
vtemp[Z] + agl	REAL	Absolute altitude.
Calls		
Function	Where Described	
copy_xy_on_tdb	Sec. 2.14.1.2.5	
tdb_get_gl	Sec. 2.14.1.2.2	

Table 2.6-144: agl_to_abs_altitude CSU [8.6.4.2.9]

2.6.4.2.10 dir_to_tan_point CSU

This CSU determines the direction from a point to a tangent point on a circle. It returns TRUE if *start* is outside the circle and *point* is valid, FALSE otherwise.

Parameters		
Parameters	Type	Where Typedef Declared
center	VECTOR	sim_types.h
start	VECTOR	sim_types.h
point	VECTOR	sim_types.h
radius	REAL	sim_types.h
radius2	REAL	sim_types.h
dtopoint	REAL	sim_types.h
ReturnValues		
Return Value	Type	Meaning
TRUE	int	<i>start</i> is outside circle and <i>point</i> is valid.
FALSE	int	<i>start</i> is not outside circle or <i>point</i> is invalid.

Calls	
Function	Where Described
vec2_sub	Sec. 2.14.3.5.23
vec2_mag2	Sec. 2.14.3.5.27
vec2_set	Sec. 2.14.3.5.21
vec2_scale	Sec. 2.14.3.5.28
vec2_rot90minus	Sec. 2.14.3.5.33
vec2_add	Sec. 2.14.3.5.22

Table 2.6-145: dir_to_tan_point CSU [8.6.4.2.10]

2.6.4.2.11 waypoint_vel CSU

This multipurpose CSU determines if the current waypoint has been reached, or if it cannot be reached. In either case it returns TRUE. If there is still a chance of getting closer to a point it returns FALSE and changes the desired velocity appropriately.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
*point	REAL	sim_types.h
ReturnValues		
Return Value	Type	Meaning
TRUE	int	Waypoint either has been or cannot be reached.
FALSE	int	Waypoint can be approached more closely.
Calls		
Function	Where Described	
within_delta	Sec. 2.14.1.2.10	
vec2_scale	Sec. 2.14.3.5.28	
pilot_get_speed	Sec. 2.6.4.2.67	
vec2_sub	Sec. 2.14.3.5.23	
vec2_mag2	Sec. 2.14.3.5.27	
interior_angle_between_vectors	Sec. 2.14.3.5.8	
vec2_norm	Sec. 2.14.3.5.31	
vec_scale	Sec. 2.6.2.64.1 Vehicles CSCI SDD	

Table 2.6-146: waypoint_vel CSU [8.6.4.2.11]

2.6.4.2.12 orbit_velocity CSU

This CSU calculates the velocity to go around a circle. A positive orbit speed gives counterclockwise turns.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
Calls		
Function	Where Described	
vec2_sub	Sec. 2.14.3.5.23	
vec2_mag2	Sec. 2.14.3.5.27	
vec2_set	Sec. 2.14.3.5.21	
vec2_norm	Sec. 2.14.3.5.31	
vec2_scale	Sec. 2.14.3.5.28	
DEBUG_PILOT	Sec. 2.5.2.2	
vec2_rot90	Sec. 2.14.3.5.32	
vec2_add	Sec. 2.14.3.5.22	

Table 2.6-147: orbit_velocity CSU [8.6.4.2.12]

2.6.4.2.13 z_velocity CSU

This CSU calculates the Z velocity to maintain an altitude above ground level.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
altitude	REAL	sim_types.h
Calls		
Function	Where Described	
min	Sec. 2.13.3.5 See Appendix A	
sign		
abs	Sec. 2.6.7.3 & Sec. 2.13.3.2 See Appendix A	
max	Sec. 2.13.3.5 See Appendix A	
vec2_scale	Sec. 2.14.3.5.28	
copy_xy_on_tdb	Sec. 2.14.1.2.5	
tdb_get_zl	Sec. 2.14.1.2.3	
vec_add	Sec. 2.6.2.57.1 Vehicles CSCI SDD	
noa_damp	Sec. 2.14.1.2.7	

Table 2.6-148: z_velocity CSU [8.6.4.2.13]

2.6.4.2.14 combined_velocity CSU

This CSU resolves the desired xy and z velocities to get a combined velocity.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
Calls		
Function	Where Described	
vec_mag3	sim_macros.h	

Table 2.6-149: combined_velocity CSU [8.6.4.2.14]

2.6.4.2.15 pilot_on_same_route CSU

This CSU determines if another pilot is on my route. If his point is found on the route out from my point, the number of points ahead or behind is placed in *num_pnt_ahead* and TRUE is returned. The route can be the same but his point may not be found if points were added to the route after the route started.

Parameters		
Parameters	Type	Where Typedef Declared
*my_point	pointer to ROUTEPOINT	Sec. 2.10.2.5
*his_point	pointer to ROUTEPOINT	Sec. 2.10.2.5
*num_pnt_ahead	int	Standard
ReturnValues		
Return Value	Type	Meaning
TRUE	int	Other pilot on same route.
FALSE	int	Other pilot not on same route.

Table 2.6-150: pilot_on_same_route CSU [8.6.4.2.15]

2.6.4.2.16 pilot_is_facing_direction CSU

This CSU returns TRUE if the pilot is facing *direction* within the *error_allowed*.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
direction	VECTOR	sim_types.h
error allowed	REAL	sim_types.h
ReturnValues		
Return Value	Type	Meaning
VEC_SMALLER2(...)	int	If TRUE, pilot is facing the direction specified.

Calls	
Function	Where Described
vec2_copy	Sec. 2.14.3.5.29
OBJ_DIRECTION	Sec. 2.9.1.1 See Appendix A
vec2_sub	Sec. 2.14.3.5.23
VEC_SMALLER2	Sec. 2.14.3.9 See Appendix A

Table 2.6-151: pilot_is_facing_direction CSU [8.6.4.2.16]

2.6.4.2.17 vehicle_facing_point CSU

This CSU determines if the vehicle is facing the *point*; that is, not turning toward it .

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.91.1
point	VECTOR	sim_types.h
ReturnValues		
Return Value	Type	Meaning
pilot_is_facing_direction(...)	int	If TRUE, pilot is facing the point.
Calls		
Function	Where Described	
vec2_sub	Sec. 2.14.3.5.23	
OBJ_POSITION	Sec. 2.9.1.1 See Appendix A	
vec2_norm	Sec. 2.14.3.5.31	
pilot_is_facing_direction	Sec. 2.6.4.2.16	

Table 2.6-152: vehicle_facing_point CSU [8.6.4.2.17]

2.6.4.2.18 are_we_there CSU

This CSU determines if a vehicle has arrived at a specified point.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*point	pointer to REAL	sim_types.h
*position	pointer to REAL	sim_types.h
speed	REAL	sim_types.h
tolerance2	REAL	sim_types.h
ReturnValues		
Return Value	Type	Meaning
TRUE	int	Arrived within tolerance2.
FALSE	int	Not arrived within tolerance2.

Calls	
Function	Where Described
vec2_sub	Sec. 2.14.3.5.23
vec2_mag2	Sec. 2.14.3.5.27
vec_init	Sec. 2.6.2.61.1 Vehicles CSCI SDD
pilot_init_land	Sec. 2.6.4.2.39

Table 2.6-153: are_we_there CSU [8.6.4.2.18]

2.6.4.2.19 xy_dir_and_range CSU

This CSU determines the xy direction and range of *p*. It assumes that *p* has pos_cur, pos_ref, and epsilon2 set.

Parameters		
Parameters	Type	Where Typedef Declared
*p	pointer to XYZ	Sec. 2.6.4.7
ReturnValues		
Return Value	Type	Meaning
TRUE	int	Range is less than the allowed epsilon.
FALSE	int	Range is greater than the allowed epsilon.
Calls		
Function	Where Described	
vec2_sub	Sec. 2.14.3.5.23	
vec2_scale	Sec. 2.14.3.5.28	

Table 2.6-154: xy_dir_and_range CSU [8.6.4.2.19]

2.6.4.2.20 compute_situation CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
*ref	pointer to REAL	sim_types.h
epsilon2	REAL	sim_types.h
ReturnValues		
Return Value	Type	Meaning
xy_dir_and_range(...)	int	If TRUE, direction and range are within the allowed epsilon.

Calls	
Function	Where Described
vec2_copy	Sec. 2.14.3.5.29
OBJ POSITION	Sec. 2.9.1.1 See Appendix A
xy_dir and range	Sec. 2.6.4.2.19

Table 2.6-155: compute_situation CSU [8.6.4.2.20]

2.6.4.2.21 project_point CSU

Parameters		
Parameters	Type	Where Typedef Declared
*headpoint	pointer to REAL	sim_types.h
*tailpoint	pointer to REAL	sim_types.h
distance	REAL	simj_types.h
*newpoint	pointer to REAL	sim_types.h
ReturnValues		
Return Value	Type	Meaning
TRUE	int	Direction and range are within the allowed epsilon.
FALSE	int	Direction and range are not within the allowed epsilon.
Calls		
Function	Where Described	
vec2_copy	Sec. 2.14.3.5.29	
xy_dir and range	Sec. 2.6.4.2.19	
vec2_scale	Sec. 2.14.3.5.28	
vec2_add	Sec. 2.14.3.5.22	

Table 2.6-156: project_point CSU [8.6.4.2.21]

2.6.4.2.22 vec2_rotate CSU

Parameters		
Parameters	Type	Where Typedef Declared
costheta	REAL	sim_types.h
sintheta	REAL	sim_types.h
*v	pointer to REAL	sim_types.h
*r	pointer to REAL	sim_types.h
Calls		
Function	Where Described	
vec2_copy	Sec. 2.14.3.5.29	

Table 2.6-157: vec2_rotate CSU [8.6.4.2.22]

2.6.4.2.23 get_next_circle_point CSU

This CSU determines the next circle point.

Parameters		
Parameters	Type	Where Typedef Declared
*mypos	pointer to REAL	sim_types.h
*center	pointer to REAL	sim_types.h
radius	REAL	sim_types.h
*point	pointer to REAL	sim_types.h
ReturnValues		
Return Value	Type	Meaning
TRUE	int	Direction and range are within the allowed epsilon.
FALSE	int	Direction and range are not within the allowed epsilon.
Calls		
Function	Where Described	
vec2 copy	Sec. 2.14.3.5.29	
xy dir and range	Sec. 2.6.4.2.19	
vec2 scale	Sec. 2.14.3.5.28	
vec2 rotate	Sec. 2.6.4.2.22	
vec2 add	Sec. 2.14.3.5.22	

Table 2.6-158: get_next_circle_point CSU [8.6.4.2.23]

2.6.4.2.24 pilot_start_takingoff CSU

This CSU causes an air vehicle to take off.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR SM	Sec. 2.6.4.7
Errors		
Error Name	Reason for Error	
ERROR_ABORT	Pilot type is neither a helicopter nor a plane.	
Calls		
Function	Where Described	
DEBUG_PILOT	Sec. 2.5.2.2 See Appendix A	
ground_level	Sec. 2.6.4.2.8	
vec_set	Sec. 2.6.3.2.36	
pilot_start_gotopoint	Sec. 2.6.4.2.26	

Table 2.6-159: pilot_start_takingoff CSU [8.6.4.2.24]

2.6.4.2.25 pilot_takingoff CSU

This CSU returns TRUE if an air vehicle is taking off and returns FALSE otherwise.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
ReturnValues		
Return Value	Type	Meaning
TRUE	int	air vehicle taking off
FALSE	int	not taking off
Calls		
Function	Where Described	
vec_set	Sec. 2.6.3.2.36	

Table 2.6-160: pilot_takingoff CSU [8.6.4.2.25]

2.6.4.2.26 pilot_start_gotopoint CSU

This CSU causes an air vehicle to go to a specified point.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
*point	pointer to REAL	sim_types.h
Calls		
Function	Where Described	
DEBUG PILOT	Sec. 2.5.2.2 See Appendix A	
OBJ POSITION	Sec. 2.9.1.1 See Appendix A	
compute_situation	Sec. 2.6.4.2.20	
vec2_copy	Sec. 2.14.3.5.29	

Table 2.6-161: pilot_start_gotopoint CSU [8.6.4.2.26]

2.6.4.2.27 pilot_gotopoint CSU

This CSU returns TRUE if an air vehicle has reached a specified point and returns FALSE otherwise.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
speed	REAL	sim_types.h
altitude	REAL	sim_types.h

ReturnValues		
Return Value	Type	Meaning
TRUE	int	Reached or passed the point.
FALSE	int	Not there yet.
Calls		
Function	Where Described	
p_follower_leader_passed_point	Sec. 2.6.4.3.14	
OBJ POSITION	Sec. 2.9.1.1 See Appendix A	
compute_situation	Sec. 2.6.4.2.20	
vec2_dot	Sec. 2.14.3.5.24	
vec2_scale	Sec. 2.14.3.5.28	
z_velocity	Sec. 2.6.4.2.13	
combined_velocity	Sec. 2.6.4.2.14	

Table 2.6-162: pilot_gotopoint CSU [8.6.4.2.27]

2.6.4.2.28 pilot_goto_endpoint CSU

This CSU returns TRUE if the vehicle is at an endpoint and returns FALSE otherwise.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR SM	Sec. 2.6.4.7
speed	REAL	sim_types.h
altitude	REAL	sim_types.h
decel	REAL	sim_types.h
ReturnValues		
Return Value	Type	Meaning
TRUE	int	At endpoint.
FALSE	int	Not at endpoint.
Calls		
Function	Where Described	
compute_situation	Sec. 2.6.4.2.20	
vec2_dot	Sec. 2.14.3.5.24	
min	Sec. 2.13.3.5 See Appendix A	
vec2_scale	Sec. 2.14.3.5.28	
z_velocity	Sec. 2.6.4.2.13	
combined_velocity	Sec. 2.6.4.2.14	

Table 2.6-163: pilot_goto_endpoint CSU [8.6.4.2.28]

2.6.4.2.29 pilot_hold CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
type	int	Standard
Errors		
Error Name	Reason for Error	
ERROR_ABORT	Hold type is neither hover, racetrack, nor orbit.	
Calls		
Function	Where Described	
pilot_init_hoverhold	Sec. 2.6.4.2.30	
pilot_init_orbithold	Sec. 2.6.4.2.34	

Table 2.6-164: pilot_hold CSU [8.6.4.2.29]

2.6.4.2.30 pilot_init_hoverhold CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR SM	Sec. 2.6.4.7
use_xy	int	Standard
x	REAL	sim_types.h
y	REAL	sim_types.h
Calls		
Function	Where Described	
pilot start gotopoint	Sec. 2.6.4.2.26	
pilot start hoverhold	Sec. 2.6.4.2.31	

Table 2.6-165: pilot_init_hoverhold CSU [8.6.4.2.30]

2.6.4.2.31 pilot_start_hoverhold CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR SM	Sec. 2.6.4.7
Calls		
Function	Where Described	
DEBUG PILOT	Sec. 2.5.2.2 See Appendix A	
pilot hoverhold	Sec. 2.6.4.2.32	

Table 2.6-166: pilot_start_hoverhold CSU [8.6.4.2.31]

2.6.4.2.32 pilot_hoverhold CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
Calls		
Function	Where Described	
OBJ_VELOCITY	Sec. 2.9.1.1 See Appendix A	
pilot is facing direction	Sec. 2.6.4.2.16	
vec_init	Sec. 2.6.2.61.1 Vehicles CSCI SDD	
z_velocity	Sec. 2.6.4.2.13	

Table 2.6-167: pilot_hoverhold CSU [8.6.4.2.32]**2.6.4.2.33 pilot_hoverhold_tick CSU**

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
Errors		
Error Name	Reason for Error	
ERROR_ABORT	Value in asm->sstate is none of the expected states.	
Calls		
Function	Where Described	
pilot_start_takingoff	Sec. 2.6.4.2.24	
pilot_takingoff	Sec. 2.6.4.2.25	
pilot_start_gotopoint	Sec. 2.6.4.2.26	
pilot_goto_endpoint	Sec. 2.6.4.2.28	
pilot_get_speed	Sec. 2.6.4.2.67	
pilot_start_hoverhold	Sec. 2.6.4.2.31	
pilot_hoverhold	Sec. 2.6.4.2.32	

Table 2.6-168: pilot_hoverhold_tick CSU [8.6.4.2.33]**2.6.4.2.34 pilot_init_orbithold CSU**

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
use_xy	int	Standard
x	REAL	sim_types.h
y	REAL	sim_types.h

Calls	
Function	Where Described
pilot_start_gotopoint	Sec. 2.6.4.2.26

Table 2.6-169: pilot_init_orbithold CSU [8.6.4.2.34]

2.6.4.2.35 pilot_start_orbithold CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
Calls		
Function	Where Described	
DEBUG_PILOT	Sec. 2.5.2.2 See Appendix A	

Table 2.6-170: pilot_start_orbithold CSU [8.6.4.2.35]

2.6.4.2.36 pilot_orbithold CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
Calls		
Function	Where Described	
get next circle point	Sec. 2.6.4.2.23	
vec copy	Sec. 2.6.2.59.1 Vehicles CSCI SDD	
vec2 copy	Sec. 2.14.3.5.29	
xy_dir and range	Sec. 2.6.4.2.19	
vec scale	Sec. 2.6.2.64.1 Vehicles CSCI SDD	
z velocity	Sec. 2.6.4.2.13	
combined velocity	Sec. 2.6.4.2.14	

Table 2.6-171: pilot_orbithold CSU [8.6.4.2.36]

2.6.4.2.37 pilot_orbit_tick CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7

Errors	
Error Name	Reason for Error
ERROR_ABORT	The value of asm->sstate is none of the expected states.
Calls	
Function	Where Described
pilot start takingoff	Sec. 2.6.4.2.24
pilot takingoff	Sec. 2.6.4.2.25
pilot start gotopoint	Sec. 2.6.4.2.26
pilot gotopoint	Sec. 2.6.4.2.27
pilot get speed	Sec. 2.6.4.2.67
pilot start orbithold	Sec. 2.6.4.2.35
pilot orbithold	Sec. 2.6.4.2.36

Table 2.6-172: pilot_orbit_tick CSU [8.6.4.2.37]

2.6.4.2.38 pilot_racetrackhold CSU

Parameters		
Parameters	Type	Where Typedef Declared
*pilot	pointer to PILOT_VARS	Sec. 2.9.1.2
*asm	pointer to AIR_SM	Sec. 2.6.4.7

Table 2.6-173: pilot_racetrackhold CSU [8.6.4.2.38]

2.6.4.2.39 pilot_init_land CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
x	REAL	sim_types.h
y	REAL	sim_types.h
Calls		
Function	Where Described	
vec2 sub	Sec. 2.14.3.5.23	
INSIDE_BOX2	Sec. 2.6.4.7 See Appendix A	
DEBUG_PILOT	Sec. 2.5.2.2 See Appendix A	
pilot start gotopoint	Sec. 2.6.4.2.26	

Table 2.6-174: pilot_init_land CSU [8.6.4.2.39]

2.6.4.2.40 pilot_start_landing CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
Errors		
Error Name	Reason for Error	
ERROR_ABORT	Pilot type is neither helicopter nor plane.	
Calls		
Function	Where Described	
DEBUG_PILOT	Sec. 2.5.2.2 See Appendix A	
vec_init	Sec. 2.6.2.61.1 Vehicles CSCI SDD	
pilot_start_landed	Sec. 2.6.4.2.42	

Table 2.6-175: pilot_start_landing CSU [8.6.4.2.40]

2.6.4.2.41 pilot_landing CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
ReturnValues		
Return Value	Type	Meaning
TRUE	int	Chopper altitude agl <= model base adjustment.
FALSE	int	Anything else.
Errors		
Error Name	Reason for Error	
ERROR_ABORT	Pilot type is other than helicopter .	
Calls		
Function	Where Described	
vec set	Sec. 2.6.3.2.36	

Table 2.6-176: pilot_landing CSU [8.6.4.2.41]

2.6.4.2.42 pilot_start_landed CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7

Errors	
Error Name	Reason for Error
ERROR_ABORT	Pilot type is neither helicopter nor plane.
Calls	
Function	Where Described
ground_level	Sec. 2.6.4.2.8
vec_init	Sec. 2.6.2.61.1 Vehicles CSCI SDD
DEBUG_PILOT	Sec. 2.5.2.2 See Appendix A
coords_within_database	Sec. 2.13.3.1 See Appendix A
tdb_place_vehicle	Sec. 2.21.7.20.5
s_atan2	Sec. 2.14.3.9 See Appendix A
report_error_from_tdb_once	Sec. 2.14.1.2.4
mat_rot_init	Sec. 2.6.2.47.1 Vehicles CSCI SDD

Table 2.6-177: pilot_start_landed CSU [8.6.4.2.42]

2.6.4.2.43 pilot_landhold_tick CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
Errors		
Error Name	Reason for Error	
ERROR_ABORT	Value in asm->sstate is none of the expected states.	
Calls		
Function	Where Described	
pilot_start takingoff	Sec. 2.6.4.2.24	
pilot takingoff	Sec. 2.6.4.2.25	
pilot_start gotopoint	Sec. 2.6.4.2.26	
pilot goto endpoint	Sec. 2.6.4.2.28	
pilot get speed	Sec. 2.6.4.2.67	
pilot_start landing	Sec. 2.6.4.2.40	
pilot landing	Sec. 2.6.4.2.41	
pilot_start landed	Sec. 2.6.4.2.42	

Table 2.6-178: pilot_landhold_tick CSU [8.6.4.2.43]

2.6.4.2.44 pilot_init_followroute CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
*route	pointer to ROUTE	Sec. 2.10.2.5
*routepoint	pointer to ROUTEPOINT	Sec. 2.10.2.5

Errors	
Error Name	Reason for Error
ERROR_ABORT	There is no routepoint.
Calls	
Function	Where Described
vec2_copy	Sec. 2.14.3.5.29
OBJ_POSITION	Sec. 2.9.1.1 See Appendix A
pilot_start_gotopoint	Sec. 2.6.4.2.26

Table 2.6-179: pilot_init_followroute CSU [8.6.4.2.44]

2.6.4.2.45 pilot_followroute_tick CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
Errors		
Error Name	Reason for Error	
ERROR_ABORT	Value in asm->sstate is none of the expected states.	
Calls		
Function	Where Described	
pilot_start_takingoff	Sec. 2.6.4.2.24	
pilot_takingoff	Sec. 2.6.4.2.25	
pilot_start_gotopoint	Sec. 2.6.4.2.26	
pilot_gotopoint	Sec. 2.6.4.2.27	
pilot_get_speed	Sec. 2.6.4.2.67	
p_follower_flip_in_turn	Sec. 2.6.4.3.23	
pilot_init_hoverhold	Sec. 2.6.4.2.30	
pilot_init_orbithold	Sec. 2.6.4.2.34	
pilot_hoverhold	Sec. 2.6.4.2.32	

Table 2.6-180: pilot_followroute_tick CSU [8.6.4.2.45]

2.6.4.2.46 pilot_init_hoverattack CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR SM	Sec. 2.6.4.7
popup_x	REAL	sim_types.h
popup_y	REAL	sim_types.h
target_x	REAL	sim_types.h
target_y	REAL	sim_types.h

Calls	
Function	Where Described
vec2 set	Sec. 2.14.3.5.21
project point	Sec. 2.6.4.2.21
vec copy	Sec. 2.6.2.59.1 Vehicles CSCI SDD
vec2 copy	Sec. 2.14.3.5.29
pilot_start_gotopoint	Sec. 2.6.4.2.26

Table 2.6-181: pilot_init_hoverattack CSU [8.6.4.2.46]

2.6.4.2.47 pilot_start_hoverattack_approach CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
Calls		
Function	Where Described	
pilot_start_gotopoint	Sec. 2.6.4.2.26	
DEBUG_PILOT	Sec. 2.5.2.2 See Appendix A	

Table 2.6-182: pilot_start_hoverattack_approach CSU [8.6.4.2.47]

2.6.4.2.48 pilot_hoverattack_approach CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
ReturnValues		
Return Value	Type	Meaning
TRUE	int	Direction and range are within the allowed epsilon.
FALSE	int	Direction and range are not within the allowed epsilon.
Calls		
Function	Where Described	
compute_situation	Sec. 2.6.4.2.20	
vec2 scale	Sec. 2.14.3.5.28	
pilot_get_speed	Sec. 2.6.4.2.67	
z_velocity	Sec. 2.6.4.2.13	
combined_velocity	Sec. 2.6.4.2.14	

Table 2.6-183: pilot_hoverattack_approach CSU [8.6.4.2.48]

2.6.4.2.49 pilot_start_hoverattack CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
Calls		
Function	Where Described	
DEBUG_PILOT	Sec. 2.5.2.2 See Appendix A	
vec_init	Sec. 2.6.2.61.1 Vehicles CSCI SDD	
targeting set fire_at pointair	Sec. 2.6.9.3.8	
targeting set fire_at will	Sec. 2.6.9.3.10	

Table 2.6-184: pilot_start_hoverattack CSU [8.6.4.2.49]

2.6.4.2.50 pilot_point_at_target CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
target_id	unsigned short	Standard
ReturnValues		
Return Value	Type	Meaning
TRUE	int	Locked onto target.
FALSE	int	Unable to lock onto target.
Calls		
Function	Where Described	
LOOKUP_VEHICLE	Sec. 2.9.3.2 See Appendix A	
DEBUG_PILOT	Sec. 2.5.2.2 See Appendix A	
vec_sub	Sec. 2.6.2.65.1 Vehicles CSCI SDD	
vec2_dot	Sec. 2.14.3.5.24	
vec2_mag2	Sec. 2.14.3.5.27	
vec_copy	Sec. 2.6.2.59.1 Vehicles CSCI SDD	
vec_scale	Sec. 2.6.2.64.1 Vehicles CSCI SDD	

Table 2.6-185: pilot_point_at_target CSU [8.6.4.2.50]

2.6.4.2.51 pilot_hoverattack CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7

ReturnValues		
Return Value	Type	Meaning
TRUE	int	
FALSE	int	
Calls		
Function	Where Described	
gunner_round_flying	Sec. 2.6.9.5.6	
vec_scale	Sec. 2.6.2.64.1 Vehicles CSCI SDD	
z_velocity	Sec. 2.6.4.2.13	

Table 2.6-186: pilot_hoverattack CSU [8.6.4.2.51]

2.6.4.2.52 pilot_start_hoverattack_egress CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
Calls		
Function	Where Described	
pilot_start_gotopoint	Sec. 2.6.4.2.26	
DEBUG PILOT	Sec. 2.5.2.2 See Appendix A	
targeting_set_hold_fire	Sec. 2.6.9.3.9	

Table 2.6-187: pilot_start_hoverattack_egress CSU [8.6.4.2.52]

2.6.4.2.53 pilot_start_hoverattack_complete CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
Calls		
Function	Where Described	
DEBUG PILOT	Sec. 2.5.2.2 See Appendix A	
pilot_hoverattack_complete	Sec. 2.6.4.2.54	

Table 2.6-188: pilot_start_hoverattack_complete CSU [8.6.4.2.53]

2.6.4.2.54 pilot_hoverattack_complete CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
Calls		
Function	Where Described	
vec_init	Sec. 2.6.2.61.1 Vehicles CSCI SDD	
z_velocity	Sec. 2.6.4.2.13	

Table 2.6-189: pilot_hoverattack_complete CSU [8.6.4.2.54]**2.6.4.2.55 pilot_hoverattack_tick CSU**

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
Errors		
Error Name	Reason for Error	
ERROR_ABORT	Value in asm->sstate is none of the expected states.	
Calls		
Function	Where Described	
pilot_start_takingoff	Sec. 2.6.4.2.24	
pilot_takingoff	Sec. 2.6.4.2.25	
pilot_start_gotopoint	Sec. 2.6.4.2.26	
pilot_gotopoint	Sec. 2.6.4.2.27	
pilot_get_speed	Sec. 2.6.4.2.67	
pilot_start_hoverattack_approach	Sec. 2.6.4.2.47	
pilot_goto_endpoint	Sec. 2.6.4.2.28	
pilot_start_hoverattack	Sec. 2.6.4.2.49	
pilot_hoverattack	Sec. 2.6.4.2.51	
pilot_start_hoverattack_egress	Sec. 2.6.4.2.52	
pilot_start_hoverattack_complete	Sec. 2.6.4.2.53	
pilot_hoverattack_complete	Sec. 2.6.4.2.54	

Table 2.6-190: pilot_hoverattack_tick CSU [8.6.4.2.55]

2.6.4.2.56 pilot_follow_vehicle CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
vid	unsigned int	Standard
xoff	int	Standard
yoff	int	Standard

Table 2.6-191: pilot_follow_vehicle CSU [8.6.4.2.56]**2.6.4.2.57 pilot_followvehicle CSU**

Parameters		
Parameters	Type	Where Typedef Declared
position	VECTOR	sim_types.h
*leadveh	pointer to SAF_OBJECT	Sec. 2.9.1.1
offset	VECTOR	sim_types.h
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
**forp	pointer to pointer to ROUTEPOINT	Sec. 2.10.2.5
*forp_dir	int	Standard
Calls		
Function	Where Described	
get_unit_direction	Sec. 2.6.3.2.11	
vec2_scale	Sec. 2.14.3.5.28	
stationpoint	Sec. 2.6.3.2.10	
vec2_sub	Sec. 2.14.3.5.23	
vec2_dot	Sec. 2.14.3.5.24	
min	Sec. 2.13.3.5 See Appendix A	
vec2_add	Sec. 2.14.3.5.22	
stop	Sec. 2.6.3.2.5	
vec_scale	Sec. 2.6.2.59.1 Vehicles CSCI SDD	
vec_copy	Sec. 2.6.2.61.1 Vehicles CSCI SDD	
set_speed_dir	Sec. 2.6.3.2.6	

Table 2.6-192: pilot_followvehicle CSU [8.6.4.2.57]**2.6.4.2.58 pilot_face_direction CSU**

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
mathradians	REAL	sim_types.h

Calls	
Function	Where Described
vec_set	Sec. 2.6.3.2.36

Table 2.6-193: pilot_face_direction CSU [8.6.4.2.58]

2.6.4.2.59 pilot_tick2 CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
ReturnValues		
Return Value	Type	Meaning
TRUE	int	Pilot state idle or unknown.
FALSE	int	Pilot state not idle and not unknown.
Errors		
Error Name	Reason for Error	
ERROR_ABORT	Value in asm->pstate is none of the expected states.	
Calls		
Function	Where Described	
pilot_hoverhold tick	Sec. 2.6.4.2.33	
pilot_orbit tick	Sec. 2.6.4.2.37	
pilot_landhold tick	Sec. 2.6.4.2.42	
pilot_followroute tick	Sec. 2.6.4.2.45	
pilot_follow_leader tick	Sec. 2.6.4.3.22	
pilot_hoverattack tick	Sec. 2.6.4.2.55	

Table 2.6-194: pilot_tick2 CSU [8.6.4.2.59]

2.6.4.2.60 idle_tick CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
Errors		
Error Name	Reason for Error	
ERROR_ABORT	Pilot type is neither helicopter nor plane.	

Calls	
Function	Where Described
pilot landing	Sec. 2.6.4.2.41
pilot start landed	Sec. 2.6.4.2.42
are we there	Sec. 2.6.4.2.18
pilot get speed	Sec. 2.6.4.2.67
pilot start landing	Sec. 2.6.4.2.40

Table 2.6-195: idle_tick CSU [8.6.4.2.60]

2.6.4.2.61 attackatwill_tick CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
ReturnValues		
Return Value	Type	Meaning
TRUE	int	Pilot state is hover attack, point at target is called, and target is valid.
FALSE	int	TRUE condition not extant.
Calls		
Function	Where Described	
RANGE CLIP	Sec. 2.14.3.9 See Appendix A	
pilot get speed	Sec. 2.6.4.2.67	
gunner round flying	Sec. 2.6.9.5.6	
vec scale	Sec. 2.6.2.64.1 Vehicles CSCS SDD	
z velocity	Sec. 2.6.4.2.13	

Table 2.6-196: attackatwill_tick CSU [8.6.4.2.61]

2.6.4.2.62 attackatwill_tick_new CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
ReturnValues		
Return Value	Type	Meaning
TRUE	int	
FALSE	int	
Errors		
Error Name	Reason for Error	
ERROR_ABORT	Value in pilot->fireat_state is none of the expected states.	

Calls	
Function	Where Described
RANGE CLIP	Sec. 2.14.3.9 See Appendix A
pilot get speed	Sec. 2.6.4.2.67
gunner round flying	Sec. 2.6.9.5.6
vec scale	Sec. 2.6.2.64.1 Vehicles CSCI SDD
z velocity	Sec. 2.6.4.2.13

Table 2.6-197: attackatwill_tick_new CSU [8.6.4.2.62]

2.6.4.2.63 pilot_check_state CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
DEBUG STATION	Sec. 2.5.2.2 See Appendix A	
OBJ VEHICLEID	Sec. 2.9.1.1 See Appendix A	

Table 2.6-198: pilot_check_state CSU [8.6.4.2.63]

2.6.4.2.64 pilot_tick CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
pilot check state	Sec. 2.6.4.2.63	
attackatwill tick	Sec. 2.6.4.2.61	
pilot tick2	Sec. 2.6.4.2.59	
idle tick	Sec. 2.6.4.2.60	
vec copy	Sec. 2.6.2.59.1 Vehicles CSCI SDD	

Table 2.6-199: pilot_tick CSU [8.6.4.2.64]

2.6.4.2.65 pilot_get_altitude CSU

This CSU returns the currently used altitude.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1

ReturnValues		
Return Value	Type	Meaning
safobj->pilot->im.altitude	REAL	Immediate altitude.
safobj->pilot->mis.altitude	REAL	Mission altitude.

Table 2.6-200: pilot_get_altitude CSU [8.6.4.2.65]

2.6.4.2.66 pilot_change_altitude_im CSU

This CSU changes the pilot's altitude immediately.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
altitude	REAL	sim_types.h
type	int	Standard

Table 2.6-201: pilot_change_altitude_im CSU [8.6.4.2.66]

2.6.4.2.67 pilot_get_speed CSU

This CSU gives the currently used speed.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
ReturnValues		
Return Value	Type	Meaning
safobj->airveh->max_mps_forward	REAL	Maximum allowed forward speed.
safobj->airveh->min_mps_forward	REAL	Minimum allowed forward speed.
speed	REAL	Current speed.

Table 2.6-202: pilot_get_speed CSU [8.6.4.2.67]

2.6.4.2.68 pilot_change_speed_im CSU

This CSU changes the pilot's speed immediately.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
speed	REAL	sim_types.h

Table 2.6-203: pilot_change_speed_im CSU [8.6.4.2.68]

2.6.4.2.69 pilot_goto_point_im CSU

This CSU moves the airvehicle to a point immediately.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
x	REAL	sim_types.h
y	REAL	sim_types.h
Errors		
Error Name	Reason for Error	
ERROR_ABORT	Safobj pilot type is neither a helicopter nor a plane.	
Calls		
Function	Where Described	
pilot_init_hoverhold	Sec. 2.6.4.2.30	

Table 2.6-204: pilot_goto_point_im CSU [8.6.4.2.69]

2.6.4.2.70 pilot_land_im CSU

This CSU causes the pilot to land immediately.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
x	REAL	sim_types.h
y	REAL	sim_types.h
Calls		
Function	Where Described	
pilot_init_land	Sec. 2.6.4.2.39	

Table 2.6-205: pilot_land_im CSU [8.6.4.2.70]

2.6.4.2.71 pilot_cancel_immediate CSU

This CSU puts the pilot into an idle state immediately.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1

Table 2.6-206: pilot_cancel_immediate CSU [8.6.4.2.71]

2.6.4.2.72 pilot_executing_immediate_command CSU

This CSU takes the pilot out of the idle state immediately.

Parameters		
Parameters	Type	Where Typedef Declared
*pilot	pointer to PILOT_VARS	
ReturnValues		
Return Value	Type	Meaning
pilot->im.pstate != PSTATE_IDLE	int	Pilot state set to not idle.

Table 2.6-207: pilot_executing_immediate_command CSU [8.6.4.2.72]

2.6.4.2.73 pilot_follow_vehicle_im CSU

This CSU has a pilot immediately follow a vehicle.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
vid	unsigned int	Standard
xoff	int	Standard
yoff	int	Standard
Calls		
Function	Where Described	
pilot_follow_vehicle	Sec. 2.6.4.2.56	

Table 2.6-208: pilot_follow_vehicle_im CSU [8.6.4.2.73]

2.6.4.2.74 pilot_face_direction_im CSU

This CSU has the pilot immediately face in a specified direction.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
mathradians	REAL	sim_types.h
Calls		
Function	Where Described	
pilot_face_direction	Sec. 2.6.4.2.58	

Table 2.6-209: pilot_face_direction_im CSU [8.6.4.2.74]

2.6.4.2.75 pilot_hoverattack_im CSU

This CSU initiates an immediate hoverattack by the pilot.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
popup_x	REAL	sim_types.h
popup_y	REAL	sim_types.h
target_x	REAL	sim_types.h
target_y	REAL	sim_types.h
Calls		
Function	Where Described	
pilot_init_hoverattack	Sec. 2.6.4.2.46	

Table 2.6-210: pilot_hoverattack_im CSU [8.6.4.2.75]

2.6.4.2.76 pilot_hold_im CSU

This CSU has the pilot hold immediately.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
type	int	Standard
Calls		
Function	Where Described	
pilot_hold	Sec. 2.6.4.2.29	

Table 2.6-211: pilot_hold_im CSU [8.6.4.2.76]

2.6.4.2.77 pilot_mission_completed CSU

Parameters		
Parameters	Type	Where Typedef Declared
*pilot	pointer to PILOT_VARS	Sec. 2.9.1.2
ReturnValues		
Return Value	Type	Meaning
TRUE	int	Mission completed.
FALSE	int	Mission not completed.

Table 2.6-212: pilot_mission_completed CSU [8.6.4.2.77]

2.6.4.2.78 pilot_stationpoint CSU

Parameters		
Parameters	Type	Where Typedef Declared
*pilot	pointer to PILOT_VARS	Sec. 2.9.1.2
point	VECTOR	sim_types.h
ReturnValues		
Return Value	Type	Meaning
TRUE	int	Success.
FALSE	int	Not a lead vehicle.
Calls		
Function	Where Described	
stationpoint	Sec. 2.6.3.2.10	

Table 2.6-213: pilot_stationpoint CSU [8.6.4.2.78]

2.6.4.2.79 pilot_set_leader_mis CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
vid	unsigned int	Standard
x_offset	int	Standard
y_offset	int	Standard
Errors		
Error Name	Reason for Error	
"pilot_set_leader_mis called on safobj without pilot"	No pilot.	
"Veh ... told to follow nonexistent vehicle"	No lead vehicle.	
"Veh ... cannot follow itself"	This vehicle is the lead vehicle.	
Calls		
Function	Where Described	
ERROR_OUT	Sec. 2.5.2.2 See Appendix A	
abort		
LOOKUP SAFOBJ	Sec. 2.9.1.1 See Appendix A	

Table 2.6-214: pilot_set_leader_mis CSU [8.6.4.2.79]

2.6.4.2.80 pilot_follow_leader CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1

Errors	
Error Name	Reason for Error
"pilot_set_leader_mis called on safobj without pilot"	No pilot.
"Veh ... Unable to follow, no leader"	No lead vehicle.
Calls	
Function	Where Described
ERROR_OUT	Sec. 2.5.2.2 See Appendix A
abort	

Table 2.6-215: pilot_follow_leader CSU [8.6.4.2.80]

2.6.4.2.81 pilot_set_route_mis CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*route	pointer to ROUTE	Sec. 2.10.2.5
*routept	pointer to ROUTEPOINT	Sec. 2.10.2.5
Calls		
Function	Where Described	
pilot_cancel_immediate	Sec. 2.6.4.2.71	
pilot_init_followroute	Sec. 2.6.4.2.44	

Table 2.6-216: pilot_set_route_mis CSU [8.6.4.2.81]

2.6.4.2.82 pilot_stop_mission CSU

This CSU stops the mission by setting the state idle.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1

Table 2.6-217: pilot_stop_mission CSU [8.6.4.2.82]

2.6.4.2.83 pilot_set_speed_mis CSU

This CSU sets the mission speed to *speed*.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
speed	REAL	sim types.h

ReturnValues		
Return Value	Type	Meaning
lsafobj->pilot->useimspeed	int	new speed

Table 2.6-218: pilot_set_speed_mis CSU [8.6.4.2.83]

2.6.4.2.84 pilot_get_asm CSU

Parameters		
Parameters	Type	Where Typedef Declared
*leader	pointer to SAF_OBJECT	Sec. 2.9.1.1
ReturnValues		
Return Value	Type	Meaning
&(leader->pilot->im)	pointer to AIR_SM	Immediate asm.
&(leader->pilot->mis)	pointer to AIR_SM	Mission asm.

Table 2.6-219: pilot_get_asm CSU [8.6.4.2.84]

2.6.4.2.85 pilot_execute_overlay CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*overlay	pointer to OVERLAY	Sec. 2.10.2.2
*cm	pointer to CONTROL MEASURE	Sec. 2.10.2.2
Errors		
Error Name	Reason for Error	
"Cannot execute anything but route cm"	Control measure is not of type route.	
Calls		
Function	Where Described	
ERROR_OUT	Sec. 2.5.2.2 See Appendix A	
pilot set route mis	Sec. 2.6.4.2.81	

Table 2.6-220: pilot_execute_overlay CSU [?????]

2.6.4.3 p_follower.c CSC

/simnet/src/host/p_follower.c

This CSC contains code for the pilots following the leader.

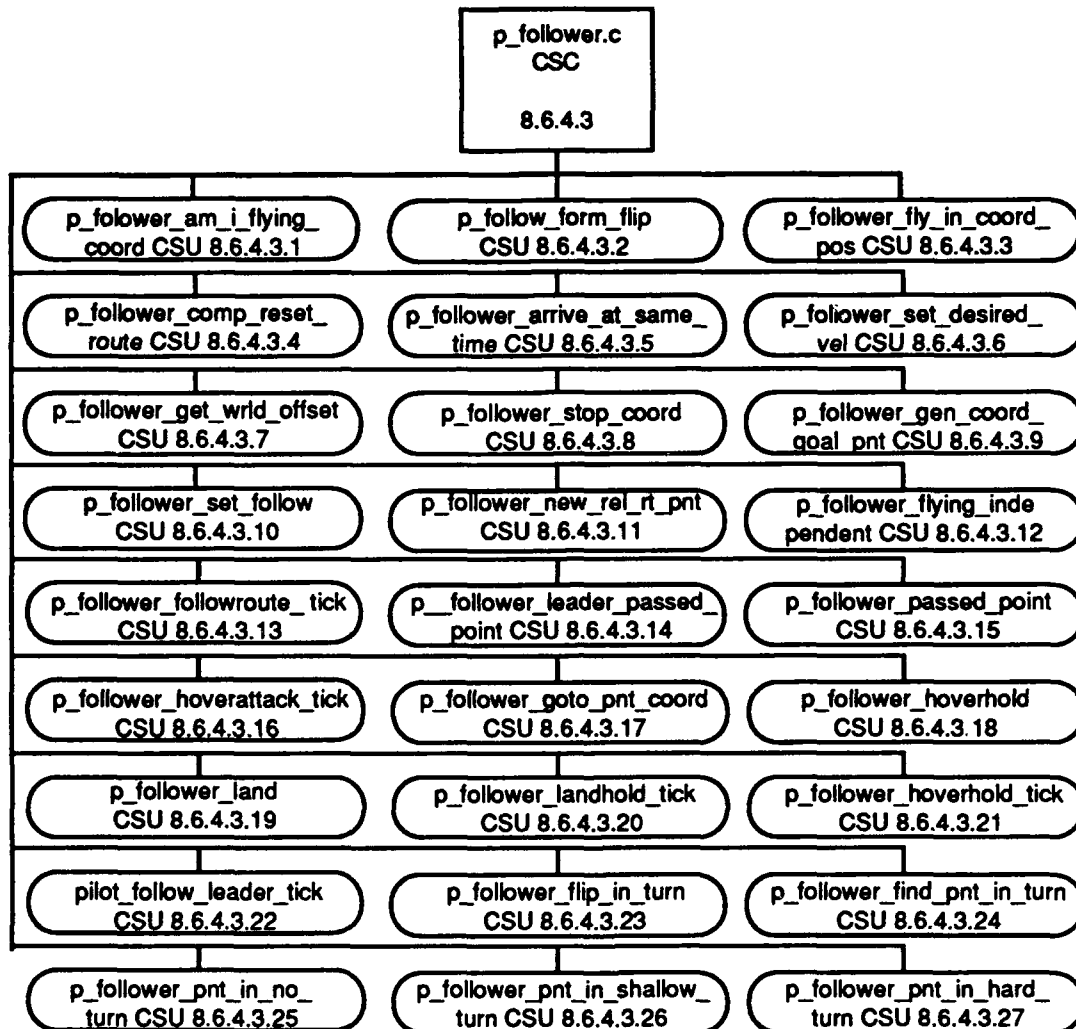


Figure 2.6-15: p_follower.c CSC Structure

2.6.4.3.1 p_follower_am_i_flying_coord CSU.

This CSU returns the logical OR results of whether goal coordinate points are set for the aircraft to determine whether a plane is flying independently or not.

Parameters		
Parameters	Type	Where Typedef Declared
*asm	pointer to AIR SM	Sec. 2.6.4.7

ReturnValues		
Return Value	Type	Meaning
(asm->coord_goal_pnt[X]) (asm->coord_goal_pnt[Y])	int	If TRUE, flight is independent.

Table 2.6-221: p_follower_am_i_flying_coord CSU [8.6.4.3.1]

2.6.4.3.2 p_follow_form_flip CSU

In this CSU if *type_of_turn*, the parameter passed, is true (-1), this is returned, indicating that the formation X positions should flip. Otherwise 1 is returned and no flip is required because there is no turn.

Parameters		
Parameters	Type	Where Typedef Declared
type_of_turn	int	Standard
ReturnValues		
Return Value	Type	Meaning
type_of_turn = -1	int	Flip formation X positions.
1	int	No flip when no turn.

Table 2.6-222: p_follow_form_flip CSU [8.6.4.3.2]

2.6.4.3.3 p_follower_fly_in_coord_pos CSU

This CSU sets the follower plane's velocity so it is flying its correct position in the formation relative to the leader's plane.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR SM	Sec. 2.6.4.7
lead_vec	VECTOR	sim_types.h
lead_est_pos	VECTOR	sim_types.h
lead_vel	REAL	sim_types.h
Calls		
Function	Where Described	
vec2_norm	Sec. 2.14.3.5.31	
vec2_scale	Sec. 2.14.3.5.28	
vec2_add	Sec. 2.14.3.5.22	
p_follower_get_wrlid_offset	Sec. 2.6.4.3.7	
vec2_sub	Sec. 2.14.3.5.23	
OBJ_POSITION	Sec. 2.9.1.1 See Appendix A	
vec2_mag	Sec. 2.14.3.5.26	

Table 2.6-223: p_follower_fly_in_coord_pos CSU [8.6.4.3.3]

2.6.4.3.4 p_follower_comp_reset_route CSU

This CSU resets the following vehicles of a composite by making them require the route from the leader.

Parameters		
Parameters	Type	Where Typedef Declared
*composite	pointer to COMPOSITE_VARS	Sec. 2.9.1.2
Calls		
Function	Where Described	
DEBUG_PILOT	Sec. 2.5.2.2 See Appendix A	

Table 2.6-224: p_follower_comp_reset_route CSU [8.6.4.3.4]

2.6.4.3.5 p_follower_arrive_at_same_time CSU

This CSU gets the following vehicle to arrive at a point at the same time that the leader arrives at his point.

Parameters		
Parameters	Type	Where Typedef Declared
*saf object	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
lead_vec	VECTOR	sim_types.h
lead_est_pos	VECTOR	sim_types.h
vec_to_goal	VECTOR	sim_types.h
Calls		
Function	Where Described	
vec2_mag	Sec. 2.14.3.5.26	
pilot_get_speed	Sec. 2.6.4.2.67	
vec2_scale	Sec. 2.14.3.5.28	

Table 2.6-225: p_follower_arrive_at_same_time CSU [8.6.4.3.5]

2.6.4.3.6 p_follower_set_desired_vel CSU

This CSU sets the velocity of the following vehicle based on the leader's velocity.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
*leader_asm	pointer to AIR_SM	Sec. 2.6.4.7
lead_est_pos	VECTOR	sim_types.h

Calls	
Function	Where Described
p_follower_am_i_flying_coord	Sec. 2.6.4.3.1
vec2_sub	Sec. 2.14.3.5.23
OBJ POSITION	Sec. 2.9.1.1 See Appendix A
p_follower_fly_in_coord_pos	Sec. 2.6.4.3.3
pilot_get_speed	Sec. 2.6.4.2.67
p_follower_arrive_at_same_time	Sec. 2.6.4.3.5
vec2_norm	Sec. 2.14.3.5.31
vec2_scale	Sec. 2.14.3.5.28
vec2_copy	Sec. 2.14.3.5.29
vec2_init	Sec. 2.14.3.5.20
z_velocity	Sec. 2.6.4.2.13
combined_velocity	Sec. 2.6.4.2.14

Table 2.6-226: p_follower_set_desired_vel CSU [8.6.4.3.6]

2.6.4.3.7 p_follower_get_wrld_offset CSU

This CSU calculates the offset point to the leader's point.

Parameters		
Parameters	Type	Where Typedef Declared
point	VECTOR	sim types.h
*asm	pointer to AIR SM	Sec. 2.6.4.7
use_u_vec	int	Standard
lead_unit_vec	VECTOR	sim types.h
lead_est_pos	VECTOR	sim types.h
art_point	VECTOR	sim types.h

Calls	
Function	Where Described
vec2_copy	Sec. 2.14.3.5.29
vec2_sub	Sec. 2.14.3.5.23
vec2_norm	Sec. 2.14.3.5.31
vec2_vh2world	Sec. 2.6.3.2.37
vec2_add	Sec. 2.14.3.5.22

Table 2.6-227: p_follower_get_wrld_offset CSU [8.6.4.3.7]

2.6.4.3.8 p_follower_stop_coord CSU

This CSU stops the follower from coordinating with the leader because no room is left for intermediate coordinating points. Instead the follower is headed toward the route goal point.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR SM	Sec. 2.6.4.7
Calls		
Function	Where Described	
vec2_init	Sec. 2.14.3.5.20	
DEBUG PILOT	Sec. 2.5.2.2 See Appendix A	
vec2_copy	Sec. 2.14.3.5.29	

Table 2.6-228: p_follower_stop_coord CSU [8.6.4.3.8]

2.6.4.3.9 p_follower_gen_coord_goal_pnt CSU

This CSU generates the new coordinate goal position. It also decides if the vehicle is about to do a sharp turn, and sets the coordinate goal point to zero so the vehicle will fly on its own.

Parameters		
Parameters	Type	Where Typedef Declared
*sabobj	pointer to SAF OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR SM	Sec. 2.6.4.7
*leader asm	pointer to AIR SM	Sec. 2.6.4.7
lead_est_pos	VECTOR	sim types.h
point	VECTOR	sim types.h
Calls		
Function	Where Described	
pilot get speed	Sec. 2.6.4.2.67	
p follower get wrld offset	Sec. 2.6.4.3.7	
vec2_sub	Sec. 2.14.3.5.23	
OBJ POSITION	Sec. 2.9.1.1 See Appendix A	
vec2_mag2	Sec. 2.14.3.5.27	
p_follower_stop_coord	Sec. 2.6.4.3.8	
vec2_mag	Sec. 2.14.3.5.26	
vec2_scale	Sec. 2.14.3.5.28	
DEBUG PILOT	Sec. 2.5.2.2 See Appendix A	

Table 2.6-229: p_follower_gen_coord_goal_pnt CSU [8.6.4.3.9]

2.6.4.3.10 p_follower_set_follow CSU

This CSU checks a vehicle and if it is not the lead vehicle sets it to follow.

Parameters		
Parameters	Type	Where Typedef Declared
*pilot	pointer to PILOT_VARS	Sec. 2.9.1.2

Table 2.6-230: p_follower_set_follow CSU [8.6.4.3.10]

2.6.4.3.11 p_follower_new_rel_rt_pnt CSU

This CSU generates a new relative route point.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
*leader_asm	pointer to AIR_SM	Sec. 2.6.4.7
indep	int	Standard
Calls		
Function	Where Described	
vec2_copy	Sec. 2.14.3.5.29	
OBJ POSITION	Sec. 2.9.1.1 See Appendix A	
p follow form flip	Sec. 2.6.4.3.2	
p follower find pnt in turn	Sec. 2.6.4.3.24	
vec2_init	Sec. 2.14.3.5.20	

Table 2.6-231: p_follower_new_rel_rt_pnt CSU [8.6.4.3.11]

2.6.4.3.12 p_follower_flying_independent CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
*leader_asm	pointer to AIR_SM	Sec. 2.6.4.7
num_pnt_ahed	int	Standard
lead_est_pos	VECTOR	sim types.h
lead_goal_point	VECTOR	sim types.h

Calls	
Function	Where Described
p_follower_passed_point	Sec. 2.6.4.3.15
OBJ POSITION	Sec. 2.9.1.1 See Appendix A
DEBUG PILOT	Sec. 2.5.2.2 See Appendix A
p_follower_new_rel_rt_pnt	Sec. 2.6.4.3.11
vehicle_facing_point	Sec. 2.6.4.2.17
p_follower_gen_coord_goal_pnt	Sec. 2.6.4.3.9
vec2_sub	Sec. 2.14.3.5.23
vec2_mag2	Sec. 2.14.3.5.27
pilot_get_speed	Sec. 2.6.4.2.67

Table 2.6-232: p_follower_flying_independent CSU [8.6.4.3.12]

2.6.4.3.13 p_follower_followroute_tick CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
*leader_asm	pointer to AIR_SM	Sec. 2.6.4.7
Errors		
Error Name	Reason for Error	
ERROR_ABORT	Value in leader_asm->sstate is none of the expected states.	
Calls		
Function	Where Described	
p_follower_hoverhold	Sec. 2.6.4.3.18	
pilot_start_takingoff	Sec. 2.6.4.2.24	
pilot_takingoff	Sec. 2.6.4.2.25	
DEBUG_PILOT	Sec. 2.5.2.2 See Appendix A	
pilot_on_same_route	Sec. 2.6.4.2.15	
p_follower_new_rel_rt_pnt	Sec. 2.6.4.3.11	
saf_vehicle_est_position	Sec. 2.6.1.1.58	
p_follower_am_i_flying_coord	Sec. 2.6.4.3.1	
p_follower_flying_independent	Sec. 2.6.4.3.12	
p_follower_passed_point	Sec. 2.6.4.3.15	
OBJ_POSITION	Sec. 2.9.1.1 See Appendix A	
p_follower_stop_coord	Sec. 2.6.4.3.8	
p_follower_set_desired_vel	Sec. 2.6.4.3.6	

Table 2.6-233: p_follower_followroute_tick CSU [8.4.3.13]

2.6.4.3.14 p_follower_leader_passed_point CSU

This CSU uses the same method as `p_follower_passed_point`, which follows, for the passing of point detection except that it leaves the needed variables in the asm `p` set correctly.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*p	pointer to XYZ	Sec. 2.6.4.7
past_loc	VECTOR	sim_types.h
cur_loc	VECTOR	sim_types.h
goal	VECTOR	sim_types.h
ReturnValues		
Return Value	Type	Meaning
FALSE	int	Not passed point.
TRUE	int	Passed point.
Calls		
Function	Where Described	
vec2_sub	Sec. 2.14.3.5.23	
vec2_mag2	Sec. 2.14.3.5.27	
vec2_scale	Sec. 2.14.3.5.28	
OBJ_DIRECTION	Sec. 2.9.1.1 See Appendix A	
pilot_get_speed	Sec. 2.6.4.2.67	
vec2_dot	Sec. 2.14.3.5.24	

Table 2.6-234: p_follower_leader_passed_point CSU [8.6.4.3.14]

2.6.4.3.15 p_follower_passed_point CSU

This CSU uses the dot product to test if a point is passed.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
past_loc	VECTOR	sim_types.h
cur_loc	VECTOR	sim_types.h
goal	VECTOR	sim_types.h
ReturnValues		
Return Value	Type	Meaning
FALSE	int	Point not passed.
TRUE	int	Point passed.

Calls	
Function	Where Described
vec2 sub	Sec. 2.14.3.5.23
vec2 scale	Sec. 2.14.3.5.28
OBJ DIRECTION	Sec. 2.9.1.1 See Appendix A
pilot get speed	Sec. 2.6.4.2.67
vec2 dot	Sec. 2.14.3.5.24

Table 2.6-235: p_follower_passed_point CSU [8.4.3.15]

2.6.4.3.16 p_follower_hoverattack_tick CSU

This CSU does nothing presently, and is included because it is called elsewhere.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
*leader_asm	pointer to AIR_SM	Sec. 2.6.4.7

Table 2.6-236: p_follower_hoverattack_tick CSU [8.6.4.3.16]

2.6.4.3.17 p_follower_goto_pnt_coord CSU

This CSU causes the follower to coordinate with the leader on the immediate goto point.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
*leader_asm	pointer to AIR_SM	Sec. 2.6.4.7
Calls		
Function	Where Described	
saf vehicle est position	Sec. 2.6.1.1.58	
vec2 sub	Sec. 2.14.3.5.23	
p follower fly in coord pos	Sec. 2.6.4.3.3	
pilot get speed	Sec. 2.6.4.2.67	

Table 2.6-237: p_follower_goto_pnt_coord CSU [8.6.4.3.17]

2.6.4.3.18 p_follower_hoverhold CSU

This CSU looks at where the leader is stopping, and stops the follower in the appropriate position, taking the follower out of the "follow leader" mode.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR SM	Sec. 2.6.4.7
*leader_asm	pointer to AIR SM	Sec. 2.6.4.7
Calls		
Function	Where Described	
vec copy	Sec. 2.14.3.5.17	
OBJ DIRECTION	Sec. 2.9.1.1 See Appendix A	
saf vehicle est position	Sec. 2.6.1.1.58	
p follower fly in coord pos	Sec. 2.6.4.3.3	
z velocity	Sec. 2.6.4.2.13	
combined velocity	Sec. 2.6.4.2.14	
p follower get wrld_offset	Sec. 2.6.4.3.7	
vec set	Sec. 2.6.3.2.36	
pilot init_hoverhold	Sec. 2.6.4.2.30	

Table 2.6-238: p_follower_hoverhold CSU [8.6.4.3.18]

2.6.4.3.19 p_follower_land CSU

This CSU looks where the leader is landing and lands the follower in the appropriate position, taking the follower out of the "follow leader" mode.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR SYM	Sec. 2.6.4.7
*leader_asm	pointer to AIR SYM	Sec. 2.6.4.7
Calls		
Function	Where Described	
p follower get wrld_offset	Sec. 2.6.4.3.7	
OBJ POSITION	Sec. 2.9.1.1 See Appendix A	
OBJ DIRECTION	Sec. 2.9.1.1 See Appendix A	
DEBUG PILOT	Sec. 2.5.2.2 See Appendix A	
pilot init_land	Sec. 2.6.4.2.39	

Table 2.6-239: p_follower_land CSU [8.6.4.3.19]

2.6.4.3.20 p_follower_landhold_tick CSU

This CSU handles following the leader to land.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SYM	Sec. 2.6.4.7
*leader_asm	pointer to AIR_SYM	Sec. 2.6.4.7
Errors		
Error Name	Reason for Error	
ERROR_ABORT	Value in leader_asm->sstate is none of the expected states.	
Calls		
Function	Where Described	
pilot_start takingoff	Sec. 2.6.4.2.24	
pilot takingoff	Sec. 2.6.4.2.25	
DEBUG_PILOT	Sec. 2.5.2.2 See Appendix A	
vec2_sub	Sec. 2.14.3.5.23	
OBJ_POSTION	Sec. 2.9.1.1 See Appendix A	
saf_vehicle_est_position	Sec. 2.6.1.1.58	
p_follower_fly_in_coord_pos	Sec. 2.6.4.3.3	
pilot_get_speed	Sec. 2.6.4.2.67	
z_velocity	Sec. 2.6.4.2.13	
combined_velocity	Sec. 2.6.4.2.14	
p_follower_land	Sec. 2.6.4.3.19	

Table 2.6-240: p_follower_landhold_tick CSU [8.6.4.3.20]

2.6.4.3.21 p_follower_hoverhold_tick CSU

This CSU handles following the leader to a hover.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SYM	Sec. 2.6.4.7
*leader_asm	pointer to AIR_SYM	Sec. 2.6.4.7
Errors		
Error Name	Reason for Error	
ERROR_ABORT	Value in leader_asm->sstate is none of the expected states.	

Calls	
Function	Where Described
pilot start takingoff	Sec. 2.6.4.2.24
pilot takingoff	Sec. 2.6.4.2.25
DEBUG PILOT	Sec. 2.5.2.2 See Appendix A
vec2 sub	Sec. 2.14.3.5.23
OBJ POSITION	Sec. 2.9.1.1 See Appendix A
saf vehicle est position	Sec. 2.6.1.1.58
p follower fly in coord pos	Sec. 2.6.4.3.3
pilot get speed	Sec. 2.6.4.2.67
z velocity	Sec. 2.6.4.2.13
combined velocity	Sec. 2.6.4.2.14
p follower hoverhold	Sec. 2.6.4.3.18

Table 2.6-241: p_follower_hoverhold_tick CSU [8.6.4.3.21]

2.6.4.3.22 pilot_follow_leader_tick CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR SYM	Sec. 2.6.4.7
ReturnValues		
Return Value	Type	Type
FALSE	int	Value in leader_asm->pstate is none of the expected states.
Errors		
Error Name	Reason for Error	
ERROR_ABORT	Value in leader_asm->pstate is none of the expected states.	
Calls		
Function	Where Described	
pilot get asm	Sec. 2.6.4.2.84	
p follower hoverhold tick	Sec. 2.6.4.3.21	
pilot orbit tick	Sec. 2.6.4.2.37	
p follower landhold tick	Sec. 2.6.4.3.20	
p follower followroute tick	Sec. 2.6.4.3.13	
p follower hoverattack tick	Sec. 2.6.4.3.16	

Table 2.6-242: pilot_follow_leader_tick CSU [8.6.4.3.22]

2.6.4.3.23 p_follower_flip_in_turn CSU

This CSU is used to determine for the leader whether the turn he is currently executing is a hard or shallow turn and whether to form flip (the X coordinates) or not.

Parameters		
Parameters	Type	Where Typedef Declared
point from	VECTOR	sim_types.h
point at	VECTOR	sim_types.h
point going to	VECTOR	sim_types.h
ReturnValues		
Return Value	Type	Meaning
1	int	No formation X position flip.
-1	int	Flip formation X positions.
Calls		
Function	Where Described	
vec2_sub	Sec. 2.14.3.5.23	
vec2_norm	Sec. 2.14.3.5.31	
vec2_add	Sec. 2.14.3.5.22	
abs	Sec. 2.6.7.3 & Sec. 2.13.3.2 See Appendix A	
vec2_copy	Sec. 2.14.3.5.29	
vec2_mag2	Sec. 2.14.3.5.27	

Table 2.6-243: p_follower_flip_in_turn CSU [8.6.4.3.23]

2.6.4.3.24 p_follower_find_pnt_in_turn CSU

This CSU determines if a turn is shallow or hard and gets the appropriate point for which the vehicle is to head.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJ	Sec. 2.9.1.1
*asm	pointer to AIR_SYM	Sec. 2.6.4.7
*leader_asm	pointer to AIR_SYM	Sec. 2.6.4.7

Calls	
Function	Where Described
DEBUG PILOT	Sec. 2.5.2.2 See Appendix A
p_follower_hoverhold	Sec. 2.6.4.3.18
vec2_sub	Sec. 2.14.3.5.23
vec2_norm	Sec. 2.14.3.5.31
p_follower_pnt_in_no_turn	Sec. 2.6.4.3.25
vec2_add	Sec. 2.14.3.5.22
abs	Sec. 2.6.7.3 & Sec. 2.13.3.2 See Appendix A
vec2_copy	Sec. 2.14.3.5.29
vec2_mag2	Sec. 2.14.3.5.27
p_follower_pnt_in_hard_turn	Sec. 2.6.4.3.27
p_follower_pnt_in_shallow_turn	Sec. 2.6.4.3.26

Table 2.6-244: p_follower_find_pnt_in_turn CSU [8.6.4.3.24]

2.6.4.3.25 p_follower_pnt_in_no_turn CSU

This CSU gets the destination for a vehicle following the leader to the last point on the route.

Parameters		
Parameters	Type	Where Typedef Declared
*asm	pointer to AIR_SYM	Sec. 2.6.4.7
point	VECTOR	sim_types.h
incomming_vec	VECTOR	sim_types.h
Calls		
Function	Where Described	
p_follower_get_wrlid_offset	Sec. 2.6.4.3.7	

Table 2.6-245: p_follower_pnt_in_no_turn CSU [8.6.4.3.25]

2.6.4.3.26 p_follower_pnt_in_shallow_turn CSU

This CSU gets the destination point for a vehicle following the leader into a shallow turn. This is done by rotating the formation position half of the angle of the final turn.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SYM	Sec. 2.6.4.7
*rt_pnt	pointer to ROUTEPOINT	Sec. 2.10.2.5
perp	VECTOR	sim_types.h

Calls	
Function	Where Described
vec2_copy	Sec. 2.14.3.5.29
vec2_veh2world	Sec. 2.6.3.2.37
vec2_add	Sec. 2.14.3.5.22
DEBUG_PILOT	Sec. 2.5.2.2 See Appendix A

Table 2.6-246: p_follower_pnt_in_shallow_turn CSU [8.6.4.3.26]

2.6.4.3.27 p_follower_pnt_in_hard_turn CSU

This CSU gets the destination point for a vehicle following the leader into a sharp turn. It causes the information to end up mirror imaged along the y axis of station offsets at x=0 after the turn. The information rolls under itself during the turn.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*asm	pointer to AIR_SM	Sec. 2.6.4.7
*rt_pnt	pointer to ROUTEPOINT	Sec. 2.10.2.5
perp	VECTOR	sim_types.h
last_rt_dir	VECTOR	sim_types.h
Calls		
Function	Where Described	
vec2_scale	Sec. 2.14.3.5.28	
vec_dot_prod	Sec. 2.6.2.54.1.1 Vehicles CSCI SDD	
vec2_add	Sec. 2.14.3.5.22	
DEBUG_PILOT	Sec. 2.5.2.2 See Appendix A	

Table 2.6-247: p_follower_pnt_in_hard_turn CSU [8.6.3.27]

2.6.4.4 plane.c CSC

/simnet/src/host/plane.c

This CSC handles everything that needs to be done by a fixed wing aircraft each tick. This consists of a single CSU plane_tick, the constant definition for the maximum aircraft power,

Constant	Value
MAX_AIRCRAFT_POWER	2000000 /* Newtons */

Table 2.6-248: plane.c Maximum Aircraft Power Constant Definition

the values in three three-element arrays,

Array	Values
REAL A INERTIA[]	{ 10000.0, 1000.0, 10000.0 }
REAL A K1[]	{ 6000.0, 5000.0, 0.035 }
REAL A K2[]	{ 18000.0, 20000.0, 700000.0 }

Table 2.6-249: plane.c Array Constants

and values for six REAL variables.

Variable	Value
REAL A K4	20.0
REAL A K5	.01
REAL A K6	50000 /* REAL A K6 = 15000 */
REAL A K7	22 /* 20 */ /* Air drag */
REAL A K8	110 /* 100 */ /* Air drag */
REAL A KP	40000.0

Table 2.6-250: plane.c Assigned Variables

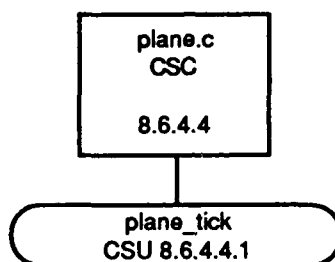


Figure 2.6-16: plane.c CSC Structure

2.6.4.4.1 plane_tick CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1

Calls	
Function	Where Described
DEBUG AIR	Sec. 2.5.2.2 See Appendix A
vec2_copy	Sec. 2.14.3.5.29
vec2_norm	Sec. 2.14.3.5.31
vec2_scale	Sec. 2.14.3.5.28
min	Sec. 2.13.3.5 See Appendix A
vec_mag3	sim macros.h
induce_roll	Sec. 2.6.4.6.11
induce_tail_spin	Sec. 2.6.4.6.10
RANGE_CLIP	Sec. 2.14.3.9 See Appendix A
max	Sec. 2.13.3.5 & Sec. 2.6.7.3 See Appendix A
square	sim macros.h
sign	
abs	Sec. 2.6.7.3 & Sec. 2.13.3.2 See Appendix A
hull_to_world_from_orientation	Sec. 2.6.4.6.8
copy_matrix_row_to_vector	Sec. 2.14.3.5.18

Table 2.6-251: plane_tick CSU [8.6.4.4.1]

2.6.4.5 impact.c CSC

/simnet/src/host/impact.c

This file handles the trajectory and impact of a round in flight.

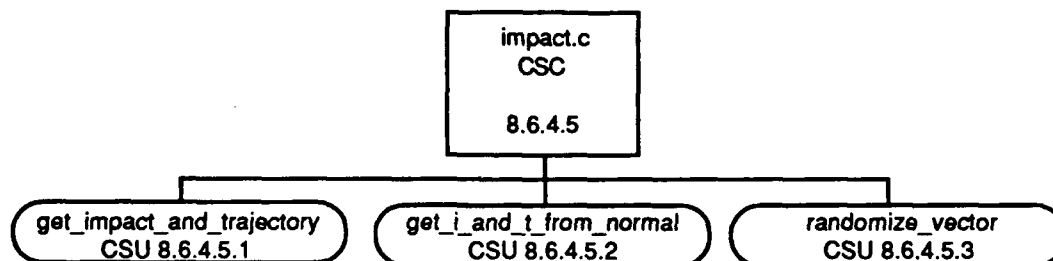


Figure 2.6-17: impact.c CSC Structure

2.6.4.5.1 get_impact_and_trajectory CSU

Parameters		
Parameters	Type	Where Typedef Declared
**victim hull to world	pointer to pointer to REAL	sim_types.h
*victim position	pointer to REAL	sim_types.h
victim turret azimuth	Angle	basic.h
victim class	VehicleClass	basic.h
*attacker position	pointer to REAL	sim_types.h
shell velocity	REAL	sim_types.h
*component	pointer to VehicleComponent	basic.h
*impact	pointer to REAL	sim_types.h
*trajectory	pointer to REAL	sim_types.h
Calls		
Function	Where Described	
vec sub	Sec. 2.6.2.65.1 Vehicles CSCI SDD	
vec normalize	Sec. 2.6.2.63.1 Vehicles CSCI SDD	
get i and t from normal	Sec. 2.2.6.4.5.2	

Table 2.6-252: get_impact_and_trajectory CSU [8.6.4.5.1]

2.6.4.5.2 get_i_and_t_from_normal CSU

Parameters		
Parameters	Type	Where Typedef Declared
**victim hull to world	pointer to pointer to REAL	sim_types.h
victim turret azimuth	Angle	basic.h
victim_class	VehicleClass	basic.h
*normal_ray_to_victim	pointer to REAL	sim_types.h
shell_velocity	REAL	sim_types.h
*component	pointer to VehicleComponent	basic.h
*impact	pointer to REAL	sim_types.h
*trajectory	pointer to REAL	sim_types.h
Calls		
Function	Where Described	
check_prob	Sec. 2.14.3.7.2	
mat rot init	Sec. 2.6.2.47.1 Vehicles CSCI SDD	
simnet_angle_to_radians	libapp.h	
mat_mat mul	Sec. 2.6.2.32.1 Vehicles CSCI SDD	
mat copy	Sec. 2.6.2.31.1 Vehicles CSCI SDD	
vec scale	Sec. 2.6.2.64.1 Vehicles CSCI SDD	
mat vec mul	Sec. 2.6.2.35.1 Vehicles CSCI SDD	
randomize_vector	Sec. 2.6.5.4.3	

Table 2.6-253: get_i_and_t_from_normal CSU [8.6.4.5.2]

2.6.4.5.3 randomize_vector CSU

Parameters		
Parameters	Type	Where Typedef Declared
*v	pointer to REAL	sim_types.h
distance	REAL	sim_types.h
*result	pointer to REAL	sim_types.h
Calls		
Function	Where Described	
get me a random fraction	Sec. 2.14.3.7.1	

Table 2.6-254: randomize_vector CSU [8.6.4.5.3]

2.6.4.6 flyingveh.c CSC

/simnet/src/host/flyingveh.c

This file contains all of the CSUs specific to air vehicles where the standard vehicle function can not apply.

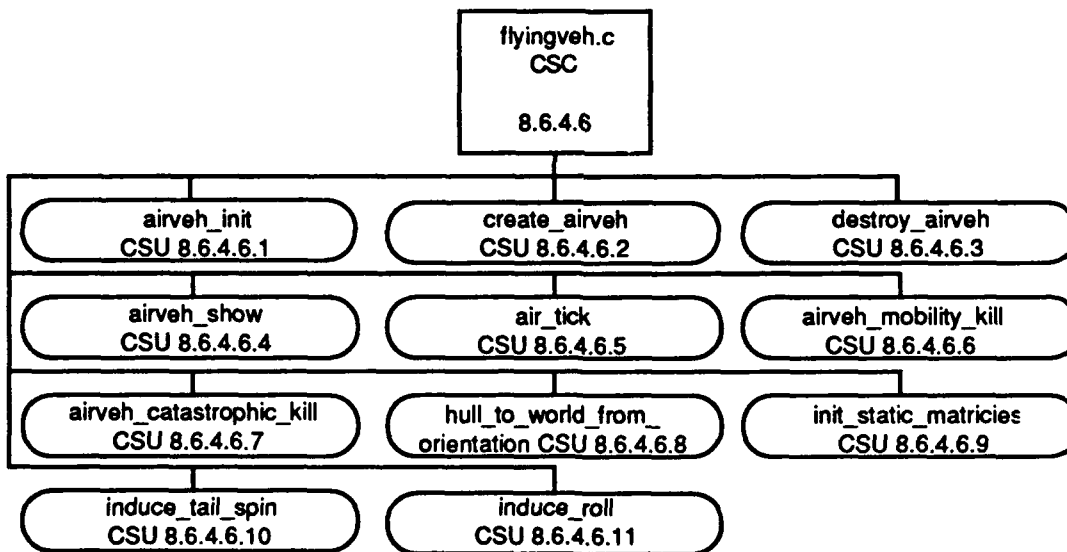


Figure 2.6-18: flyingveh.c CSC Structure

It also includes two static and constant definitions prior to init_static_matrices for the use of the remaining CSUs.

Constant	Typedef
tail spin matrix	T MATRIX
roll spin matrix	T MATRIX

Table 2.6-255: flyingvehicle.c Static Defines

Constant	Value
ROLL SPIN ROTATION RATE	-0.05
TAIL SPIN ROTATION RATE	0.05

Table 2.6-256: flyingvehicle.c Constant Definitions

2.6.4.6.1 airveh_init CSU

This CSU initializes the members of an airveh data structure.

Parameters		
Parameters	Type	Where Typedef Declared
*airveh	pointer to AIRVEH_VARS	Sec. 2.9.1.2
*table	pointer to DATA_UNION	Sec. 2.1.1.5
Calls		
Function	Where Described	
vec_init	Sec. 2.6.2.61.1 Vehicles CSCI SDD	
ft_float	Sec. 2.14.1.2.12	
deg_to_rad	sim_macros.h	
kph_to_mps	Sec. 2.13.3.1 See Appendix A	

Table 2.6-257: airveh_init CSU [8.6.4.6.1]

2.6.4.6.2 create_airveh CSU

This CSU allocates space for an airveh data structure and calls airveh_init to initialize its members.

Parameters		
Parameters	Type	Where Typedef Declared
*table	pointer to DATA_UNION	Sec. 2.1.1.5
ReturnValues		
Return Value	Type	Meaning
airveh	pointer to AIRVEH_VARS	Data structure created.
Calls		
Function	Where Described	
allocate_airveh	Sec. 2.9.1.2 See Appendix A	
airveh_init	Sec. 2.6.4.6.1	

Table 2.6-258: create_airveh CSU [8.6.4.6.2]

2.6.4.6.3 destroy_airveh CSU

This CSU deallocates memory space previously allocated for an airveh data structure, by calling the function `deallocate_airveh`.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
deallocate_airveh	Sec. 2.9.1.2	

Table 2.6-259: destroy_airveh CSU [8.6.4.6.3]

2.6.4.6.4 airveh_show CSU

This CSU displays the air vehicle variables associated with flight.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
print_vector	Sec. 2.14.3.5.2	

Table 2.6-260: airveh_show CSU [8.6.4.6.4]

2.6.4.6.5 air_tick CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
Errors		
Error Name	Reason for Error	
ERROR_ABORT	The air vehicle has crashed and it is neither a helicopter nor a plane; or the value in airveh->crashing_state is none of the expected states.	
Calls		
Function	Where Described	
helo_tick	Sec. 2.6.4.1.1	
plane_tick	Sec. 2.6.4.4.1	
ground_level	Sec. 2.6.4.2.8	
vec_init	Sec. 2.6.2.61.1 Vehicles CSCI SDD	
saf_vehicle_catastrophic_kill	Sec. 2.6.1.1.25	

Table 2.6-261: air_tick CSU [8.6.4.6.5]

2.6.4.6.6 airveh_mobility_kill CSU

This CSU sets the air vehicle crashing state to 'crashing'.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
Errors		
Error Name	Reason for Error	
ERROR_ABORT	There is no air vehicle.	

Table 2.6-262: airveh_mobility_kill CSU [8.6.4.6.6]

2.6.4.6.7 airveh_catastrophic_kill CSU

This CSU sets the air vehicle crashing state to 'crashing'.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
Errors		
Error Name	Reason for Error	
ERROR_ABORT	There is no air vehicle.	

Table 2.6-263: airveh_catastrophic_kill CSU [8.6.4.6.7]

2.6.4.6.8 hull_to_world_from_orientation CSU

Parameters		
Parameters	Type	Where Typedef Declared
orientation	VECTOR	sim_types.h
h2w	T MATRIX	sim_types.h

Table 2.6-264: hull_to_world_from_orientation CSU [8.6.4.6.8]

2.6.4.6.9 init_static_matrices CSU

This CSU initializes spin and roll matrices.

Calls	
Function	Where Described
mat_rot_init	Sec. 2.

Table 2.6-265: init_static_matrices CSU [8.6.4.6.9]

2.6.4.6.10 induce_tail_spin CSU

Parameters		
Parameters	Type	Where Typedef Declared
h2w	T MATRIX	sim_types.h
Calls		
Function	Where Described	
mat_mat_mul	Sec. 2.6.2.32.1 Vehicles CSCI SDD	

Table 2.6-266: induce_tail_spin CSU [8.6.4.6.10]**2.6.4.6.11 induce_roll CSU**

Parameters		
Parameters	Type	Where Typedef Declared
h2w	T MATRIX	sim_types.h
Calls		
Function	Where Described	
mat_mat_mul	Sec. 2.6.2.32.1 Vehicles CSCI SDD	

Table 2.6-267: induce_roll CSU [8.6.4.6.11]**2.6.4.7 flyingveh.h CSU**

/simnet/src/host/flyingveh.h

This CSU contains all of the definitions, macros, and symbolic constants needed by the flying vehicle code. The constants and structure definitions are contained in the following tables. The macros POSITION, VEHICLEID, DT, ERROR_ABORT, and INSIDE_BOX2 (point, tolerance) are described in Appendix A.

Constant	Value
CIRCLE_LOOKAHEAD_TIME	1.5
DONT_RELAND_DISTANCE	100.0
TOL2	2500.0
DECELERATION_Z	4.0
DECEL	1.5
MIN_TURN_SPEED	1.0
MAX_BACKUP_SPEED	1.0
MAX_BACKUP_DIST2	1.0

Table 2.6-268: flyingveh.h Movement Constant Definitions

Constant	Value
HELO	0
PLANE	1

Table 2.6-269: flyingveh.h Vehicle Types Constant Definitions

Constant	Value
CRASHING STATE ALIVE	0
CRASHING STATE CRASHING	1
CRASHING STATE CRASHED	2
CRASHING STATE LANDING	3
CRASHING STATE TAKINGOFF	4

Table 2.6-270: flyingveh.h Flying Vehicle Crash State Definitions

Constant	Value
PSTATE IDLE	0
PSTATE HOVERHOLD	3
PSTATE ORBITHOLD	4
PSTATE RACETRACKHOLD	5
PSTATE LANDHOLD	6
PSTATE FOLLOWROUTE	7
PSTATE FOLLOWVEHICLE	10
PSTATE FOLLOWLEADER	11
PSTATE HOVERATTACK	12
PSTATE RUNNINGATTACK	13
PSTATE FACEDIRECTION	14
SSTATE LANDED	20
SSTATE TAKINGOFF	21
SSTATE LANDING	22
SSTATE GOTOPPOINT	23
SSTATE COMPLETE	24
SSTATE APPROACH	25
SSTATE HOVERATTACK	26
SSTATE EGRESS	27
SSTATE FOLLOWLEAD	28
SSTATE FOLLOW HOVER	29

Table 2.6-271: flyingveh.h Plane and Helicopter State Constant Definitions

Constant	Value
FIRE STATE OFF	0
FIRE STATE FIRE	1
FIRE STATE RUN	2

Table 2.6-272: flyingveh.h Fire At Will States Constant Definitions

Constant	Value
NOE	0
CONTOUR	1
LOWLEVEL	2
METERSAGL	3
JINK	4
TACTICALCOLUMN	5
BOUNDING	6
BOUNDINGOVERWATCH	7
REATTACK	9
RUNNINGFIRE	10
HOVERFIRE	11

Table 2.6-273: flyingveh.h Modifiers Constant Definitions

Constant	Value
DIR POINT ERROR	.04
FWA TURN DIP ERR	.09
MAX TURN CIRCUMF	24000
INTERMEDIATE POINT TYPE	10
NO TURN	0
SHALLOW TURN	1
HARD TURN	-1
SHALLOW HARD TURN VEC MAG	3.879383241
ST KP DIST HELO	INTERMEDIATE POINT TYPE/2
ST KP DIST PLANE	INTERMEDIATE POINT TYPE

Table 2.6-274: flyingveh.h Station Keeping Constant Definitions

Constant	Value
M	0
MIS	1

Table 2.6-275: flyingveh.h State Machines Constant Definitions

The following structure is tagged xyz.

Item	Type	Where Type Defined
pos cur	VECTOR	sim_types.h
pos ref	VECTOR	sim_types.h
pos err	VECTOR	sim_types.h
dir ref	VECTOR	sim_types.h
vel des	VECTOR	sim_types.h
range	REAL	sim_types.h
range2	REAL	sim_types.h
xy_range	REAL	sim_types.h
xy_range2	REAL	sim_types.h
xy_speed des	REAL	sim_types.h
speed des	REAL	sim_types.h
angle_err	REAL	sim_types.h
epsilon2	REAL	sim_types.h

Table 2.6-276: XYZ Structure Definition

The following structure is tagged asm.

Item	Type	Where Type Defined
pstate	int	Standard
prev_pstate	int	Standard
sstate	int	Standard
prev_sstate	int	Standard
speed	REAL	sim_types.h
altitude	REAL	sim_types.h
p	XYZ	Structure before this
goto_point	VECTOR	sim_types.h
goto_start_error	VECTOR	sim_types.h
goto_range	REAL	sim_types.h
already_here	int	Standard
goto_target_speed	REAL	sim_types.h
goto_target_alt	REAL	sim_types.h
orbit_radius	REAL	sim_types.h
orbit_radius2	REAL	sim_types.h
orbit_speed	REAL	sim_types.h
orbit_center	VECTOR	sim_types.h
orbit_lookahead_time	REAL	sim_types.h
*leadveh	pointer to struct saf object	Sec. 2.9.1.1
*prev_lead	pointer to struct saf object	Sec. 2.9.1.1
stationoffset	VECTOR	sim_types.h
form_flip	int	Standard
coord_goal_pnt	VECTOR	sim_types.h
assoc_intrm_dist	REAL	sim_types.h
route_goal_pnt	VECTOR	sim_types.h
*assoc_rt_pnt	pointer to ROUTEPOINT	Sec. 2.10.2.5
*indepndnt_at	pointer to ROUTEPOINT	Sec. 2.10.2.5
point_from	VECTOR	sim_types.h
type_of_turn	int	Standard
*route	pointer to ROUTE	Sec. 2.10.2.5
*routepoint	pointer to ROUTEPOINT	Sec. 2.10.2.5
face_direction	VECTOR	sim_types.h

Table 2.6-277: AIR_SM Structure Definition

2.6.5 Intervisibility CSC

The Intervisibility CSC [8.6.5] consists of the detection.c CSC [8.6.5.1] and the intervis.c CSC [8.6.5.2]. The structure of CSC 8.6.5 is found in the following figure.

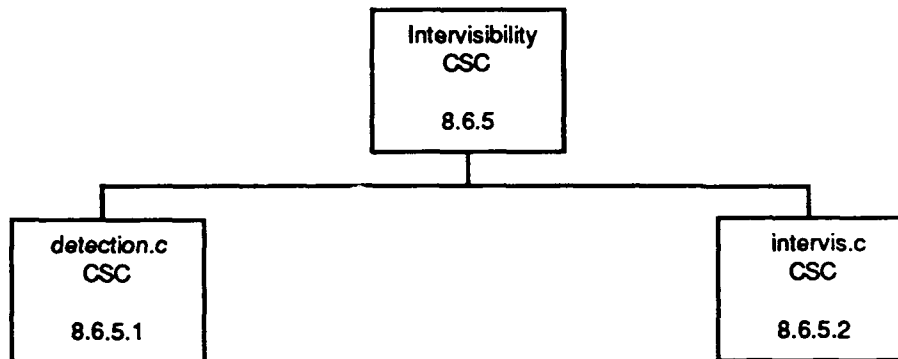


Figure 2.6-19: Intervisibility CSC Structure

2.6.5.1 detection.c CSC

/simnet/src/host/detection.c

This file handles the tracking of other vehicles, maintaining and updating the list of the other vehicles that this one currently knows about (that it has seen and still can see). This information is used to determine which vehicles will be drawn on the workstation screen, and also in maintaining this vehicle's target list.

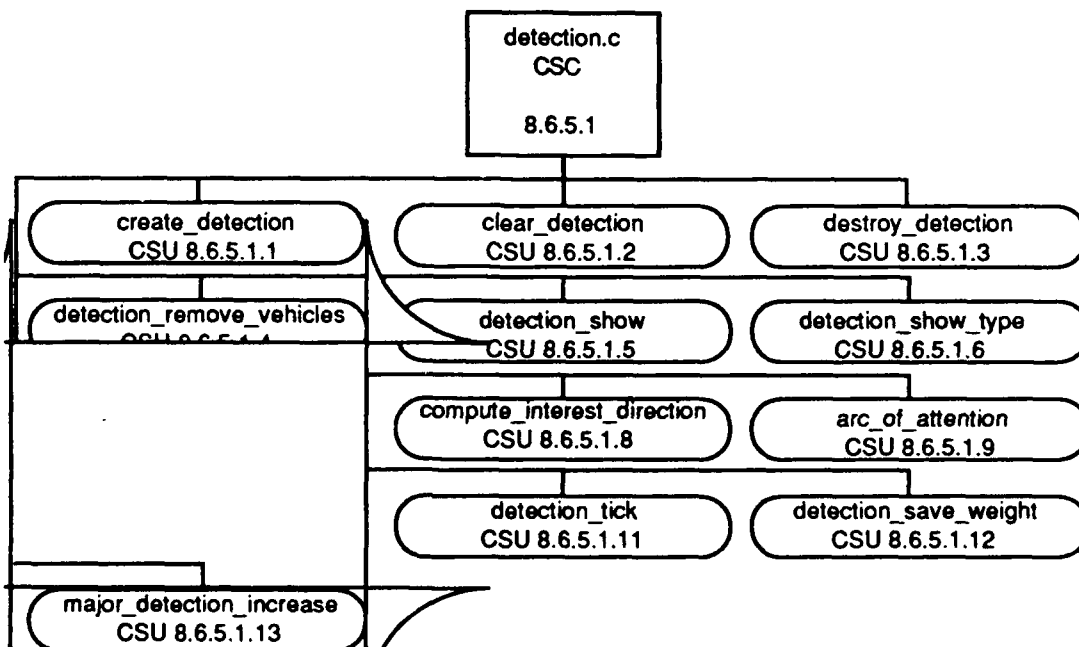


Figure 2.6-20: detection.c CSC Structure

2.6.5.1.1 create_detection CSU

This CSU creates a detection variable by allocating memory space for it.

ReturnValues		
Return Value	Type	Meaning
s	pointer to DETECTION_VARS	Sec. 2.9.1.2
Calls		
Function	Where Described	
allocate_detection	Sec. 2.9.1.2 See Appendix A	
clear_detection	Sec. 2.6.5.1.2	

Table 2.6-277: create_detection CSU [8.6.5.1.1]

2.6.5.1.2 clear_detection CSU

This CSU initializes the members of a detection data structure to zero.

Parameters		
Parameters	Type	Where Typedef Declared
*detection	pointer to DETECTION_VARS	Sec. 2.9.1.2

Table 2.6-278: clear_detection CSU [8.6.5.1.2]

2.6.5.1.3 destroy_detection CSU

This CSU deallocates a detection data structure by calling deallocate_detection.

Parameters		
Parameters	Type	Where Typedef Declared
*d	pointer to DETECTION_VARS	Sec. 2.9.1.2
Calls		
Function	Where Described	
deallocate_detection	Sec. 2.9.1.2 See Appendix A	

Table 2.6-279: destroy_detection CSU [8.6.5.1.3]

2.6.5.1.4 detection_remove_vehicles CSU

Parameters		
Parameters	Type	Where Typedef Declared
*d	pointer to DETECTION_VARS	Sec. 2.9.1.2
num	int	Standard
v_list	unsigned int	Standard

Table 2.6-280: detection_remove_vehicles CSU [8.6.5.1.4]

2.6.5.1.5 detection_show CSU

This CSU displays a listing of the vehicles arranged by detection type.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
flags	int	Standard
Calls		
Function	Where Described	
detection_show_type	Sec. 2.6.5.1.6	

Table 2.6-281: detection_show CSU [8.6.5.1.5]

2.6.5.1.6 detection_show_type CSU

This CSU displays all the vehicles having the given detection type.

Parameters		
Parameters	Type	Where Typedef Declared
tab[]	unsigned char	Standard
type	int	Standard
Calls		
Function	Where Described	
LOOKUP_VEHICLE	Sec. 2.9.3.2 See Appendix A	

Table 2.6-282: detection_show_type CSU [8.6.5.1.6]

2.6.5.1.7 compute_enemy_weight CSU

This CSU ranks enemy vehicles according to my type and their type.

Parameters		
Parameters	Type	Where Typedef Declared
*me	pointer to SAF_OBJECT	Sec. 2.9.1.1
*him	pointer to SAF_OBJECT	Sec. 2.9.1.1
ReturnValues		
Return Value	Type	Meaning
1	int	No targeting action.
16	int	I am targeting an enemy aircraft.
Calls		
Function	Where Described	
IS_AIRCRAFT		
OBJECT_TYPE		

Table 2.6-283: compute_enemy_weight CSU [8.6.5.1.7]

2.6.5.1.8 compute_interest_direction CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*vec	REAL	sim_types.h
Calls		
Function	Where Described	
turret interest_dir	Sec. 2.6.9.6.9	
vec copy	Sec. 2.6.2.59.1 Vehicles CSCI SDD	
OBJ DIRECTION	Sec. 2.9.1.1 See Appendix A	

Table 2.6-284: compute_interest_direction CSU [8.6.5.1.8]**2.6.5.1.9 arc_of_attention CSU**

Parameters		
Parameters	Type	Where Typedef Declared
angle to target	REAL	sim_types.h
ReturnValues		
Return Value	Type	Meaning
PRIMARY ARC	int	Angle to target $\leq \pi/3$
SECONDARY ARC	int	Angle to target $> \pi/3$
Calls		
Function	Where Described	
abs	Sec. 2.6.7.3 & Sec. 2.13.3.2 See Appendix A	

Table 2.6-285: arc_of_attention CSU [8.6.5.1.9]**2.6.5.1.10 detectable CSU**

This CSU returns 0 if undetectable or else a number between 1 and 99 that encodes the random number that was rolled to determine a positive detection.

Parameters		
Parameters	Type	Where Typedef Declared
*my_dir	REAL	sim_types.h
*to target	REAL	sim_types.h
*remote vel	REAL	sim_types.h
range2	REAL	sim_types.h
view_type	int	Standard

ReturnValues		
Return Value	Type	Meaning
max(1,local_rand)	int	Positive detection random number.
0	int	Undetectable.
Calls		
Function	Where Described	
vec_dot_prod	Sec. 2.6.2.6541 Vehicles CSCI SDD	
abs	Sec. 2.6.7.3 & Sec. 2.13.3.2 See Appendix A	
interior_angle_between_vectors	Sec. 2.14.3.5.8	
database_detection_query	Sec. 2.6.8.2.2	
get_me_a_random_fraction	Sec. 2.14.3.7.1	
max	Sec. 2.6.7.3 & Sec. 2.13.3.5 See Appendix A	

Table 2.6-286: detectable CSU [8.6.5.1.10]

2.6.5.1.11 detection_tick CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
OBJ_POSITION	Sec. 2.9.1.1 See Appendix A	
OBJ_FORCEID	Sec. 2.9.1.1 See Appendix A	
compute_interest_direction	Sec. 2.6.5.1.8	
FOR_VEHICLES_WITHIN_N_GRIDS_DO	Sec. 2.9.3.2 See Appendix A	
OBJ_VEHICLEID	Sec. 2.9.1.1 See Appendix A	
intervis_possibly_visible	Sec. 2.6.5.2.3	
compute_enemy_weight	Sec. 2.6.5.1.7	
detectable	Sec. 2.6.5.1.10	
OBJ_VELOCITY	Sec. 2.9.1.1 See Appendix A	

Table 2.6-287: detection_tick CSU [8.6.5.1.11]

2.6.5.1.12 detection_save_weight CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1

Table 2.6-288: detection_save_weight CSU [8.6.5.1.12]

2.6.5.1.13 major_detection_increase CSU

An increase of 40% in detection weight is a major detection increase.

Prior to this CSU, 40% value for the detection increase threshold is defined as a constant (#define DETECTION_INCREASE_THRESHOLD 40).

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
ReturnValues		
Return Value	Type	Meaning
FALSE	int	No change in enemy weight.
TRUE	int	Enemy weight increased from 0 or jumped from one level to a higher level beyond the detection increase threshold.
Calls		
Function	Where Described	
DEBUG TARGETING	Sec. 2.5.2.2 See Appendix A	

Table 2.6-289: major_detection_increase CSU [8.6.5.1.13]

2.6.5.2 intervis.c CSC

/simnet/src/host/intervis.c

This CSC contains the code which handles all the cases involving what other vehicles the current vehicle can see. Please note that this does not guarantee that another vehicle which can be seen will be seen. That is handled by the detection code.

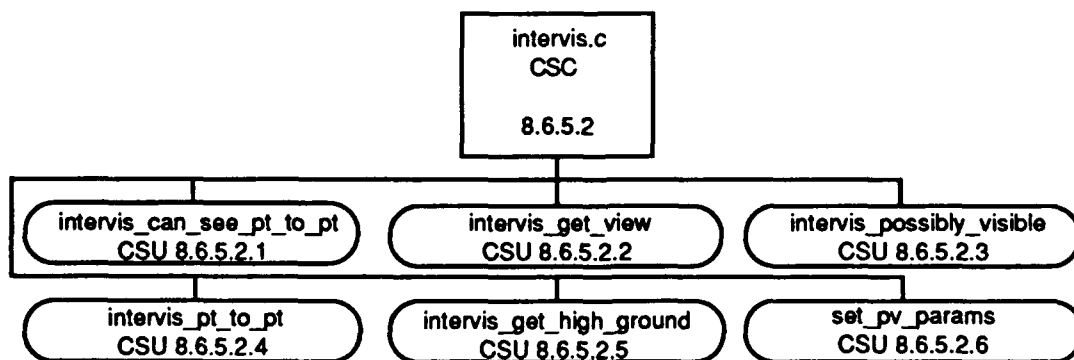


Figure 2.6-21: intervis.c CSC Structure

2.6.5.2.1 intervis_can_see_pt_to_pt CSU

This CSU checks the visibility from my point to his point and returns FALSE if it is completely blocked, TRUE if partially blocked or open.

Parameters		
Parameters	Type	Where Typedef Declared
*my_pos	pointer to REAL	sim_types.h
*his_pos	pointer to REAL	sim_types.h
ReturnValues		
Return Value	Type	Meaning
TRUE	int	Visibility is full or only partially blocked.
FALSE	int	Visibility is completely blocked.
Calls		
Function	Where Described	
intervis_pt_to_pt	Sec. 2.5.5.2.4	

Table 2.6-290: intervis_can_see_pt_to_pt CSU [8.6.5.2.1]

2.6.5.2.2 intervis_get_view CSU

Parameters		
Parameters	Type	Where Typedef Declared
*my_pos	pointer to REAL	sim_types.h
*his_pos	pointer to REAL	sim_types.h
*his_vel	pointer to REAL	sim_types.h
ReturnValues		
Return Value	Type	Meaning
HULL_DOWN	int	Visibility is partially blocked by solid or by solid and trees.
STATIONARY	int	Visibility is full or partially blocked by trees, and he is moving at a small velocity.
MOVING	int	Neither of the above two conditions holds.
VIEW_BLOCKED	int	Visibility is completely blocked.
Calls		
Function	Where Described	
intervis_pt_to_pt	Sec. 2.6.5.2.4	
vec_dot_prod	Sec. 2.6.2.54.1 Vehicles CSCI SDD	

Table 2.6-291: intervis_get_view CSU [8.6.5.2.2]

2.6.5.2.3 intervis_possibly_visible CSU

Parameters		
Parameters	Type	Where Typedef Declared
*from	pointer to REAL	sim_types.h
*to	pointer to REAL	sim_types.h
ReturnValues		
Return Value	Type	Meaning
INVISIBLE	int	The from or to coordinates are not in the database.
FALSE	int	Visibility parameter reads invisible, or unexpected results occur.
TRUE	int	Visibility parameter reads dummy or fully-visible.
Calls		
Function	Where Described	
coords within database	Sec. 2.13.3.1 See Appendix A	
set pv params	Sec. 2.6.5.2.6	
pve checkvis	Sec. 2.6.7.1.1	

Table 2.6-292: intervis_possibly_visible CSU [8.6.5.2.3]**2.6.5.2.4 intervis_pt_to_pt CSU**

Parameters		
Parameters	Type	Where Typedef Declared
*from	pointer to REAL	sim_types.h
*to	pointer to REAL	sim_types.h
ReturnValues		
Return Value	Type	Meaning
INVISIBLE	int	The from or to coordinates are not in the database.
pve checkvis(...)	int	Visibility parameter.
Calls		
Function	Where Described	
coords within database	Sec. 2.13.3.1 See Appendix A	
set pv params	Sec. 2.6.5.2.6	
pve checkvis	Sec. 2.6.7.1.1	

Table 2.6-293: intervis_pt_to_pt CSU [8.6.5.2.4]

2.6.5.2.5 intervis_get_high_ground CSU

Parameters		
Parameters	Type	Where Typedef Declared
*from	pointer to REAL	sim_types.h
*to	pointer to REAL	sim_types.h
*high_point	pointer to REAL	sim_types.h
ReturnValues		
Return Value	Type	Meaning
FALSE	int	The from or to coordinates are not in the database.
TRUE	int	Success.
Calls		
Function	Where Described	
coords_within_database	Sec. 2.13.3.1 See Appendix A	
set_pv_params	Sec. 2.6.5.2.6	
pve_checkvis	Sec. 2.6.7.1.1	

Table 2.6-294: intervis_get_high_ground CSU [8.6.5.2.5]**2.6.5.2.6 set_pv_params CSU**

Prior to this CSU, the constants S_EYE_HEIGHT and S_FOOT_HEIGHT are defined as 2.0 and 0.5 respectively.

Parameters		
Parameters	Type	Where Typedef Declared
*vis_ptr	pointer to PV_PARAMS	Sec. 2.6.7.4
from	VECTOR	sim_types.h
to	VECTOR	sim_types.h
test	int	Standard

Table 2.6-295: set_pv_params CSU [8.6.5.2.6]**2.6.6 Resupply CSC**

The Resupply CSC [8.6.6] consists of the logistics.c CSC [8.6.6.1] and the logistics.h CSC [8.6.6.2]. The structure of CSC 8.6.6 is found in the following figure.

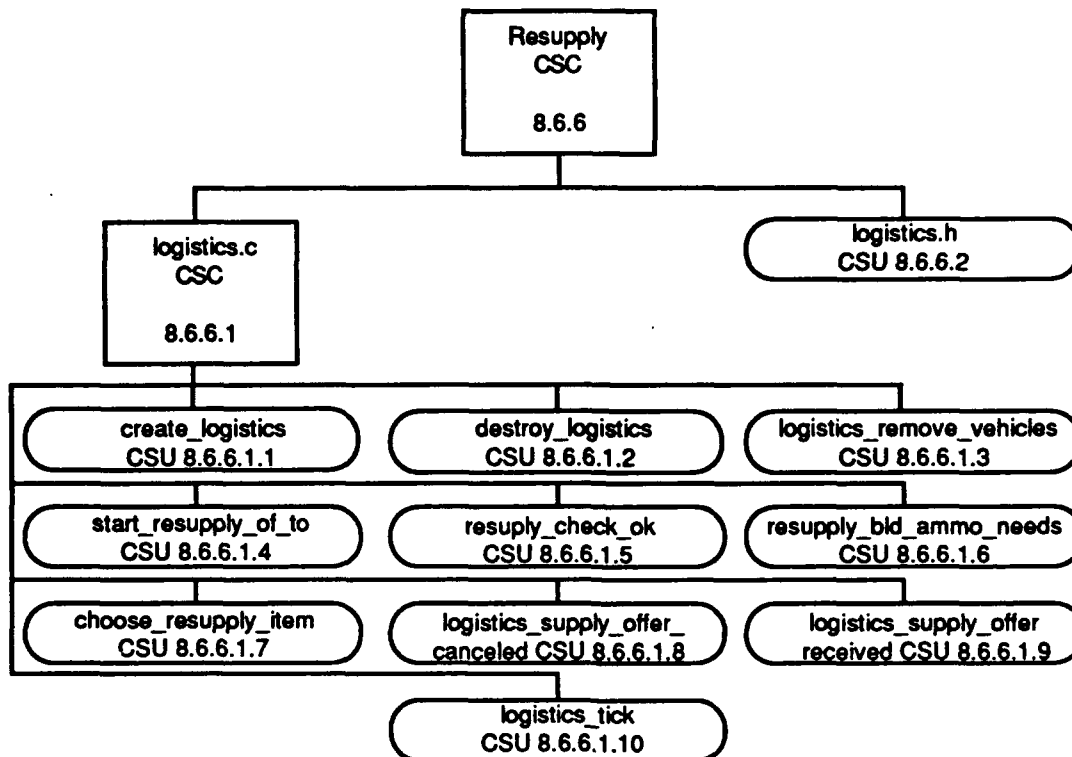


Figure 2.6-22: Logistics CSC Structure

2.6.6.1 logistics.c CSC

/simnet/src/host/logistics.c

This code handles the resupply of SAF vehicles from resupply vehicles on the database.

2.6.6.1.1 create_logistics CSU

This CSU allocates memory space for and initializes the members of a logistics data structure.

ReturnValues		
Return Value	Type	Meaning
log	int	Created logistics data structure.
Calls		
Function	Where Described	
allocate_logistics	Sec. 2.9.1.2 See Appendix A	

Table 2.6-296: create_logistics CSU [8.6.6.1.1]

2.6.6.1.2 destroy_logistics CSU

This CSU deallocates memory space that was previously reserved for a logistics data structure.

Parameters		
Parameters	Type	Where Typedef Declared
*log	pointer to LOGISTICS_VARS	Sec. 2.9.1.2
Calls		
Function	Where Described	
buffer deallocate	Sec. 2.14.4.2.15	
dealocate logistics	Sec. 2.9.1.2 See Appendix A	

Table 2.6-297: destroy_logistics CSU [8.6.6.1.2]

2.6.6.1.3 logistics_remove_vehicles CSU

If a supply vehicle is on a list provided to this CSU, it is removed from a logistics data structure.

Parameters		
Parameters	Type	Where Typedef Declared
*log	pointer to LOGISTICS_VARS	Sec. 2.9.1.2
num	int	Standard
v_list[]	unsigned int	Standard

Table 2.6-298: logistics_remove_vehicles CSU [8.6.6.1.3]

2.6.6.1.4 start_resupply_of_to CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
supplier_id	unsigned int	Standard
Calls		
Function	Where Described	
DEBUG VEHICLE	Sec. 2.5.2.2 See Appendix A	

Table 2.6-299: start_resupply_of_to CSU [8.6.6.1.4]

2.6.6.1.5 resupply_check_ok CSU

This CSU returns TRUE if conditions to resupply are met.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*supplier	pointer to SAF_OBJECT	Sec. 2.9.1.1
ReturnValues		
Return Value	Type	Meaning
TRUE	int	Conditions are met.
FALSE	int	Conditions are not met.
Calls		
Function	Where Described	
sbx_printf	Sec. 2.4.3.2.8	
OBJ_VEHICLEID	Sec. 2.9.1.1 See Appendix A	
OBJ_OWNER_PORT_NUMBER	Sec. 2.9.1.1 See Appendix A	
is dead	Sec. 2.13.3.1 See Appendix A	
OBJ_APPEARANCE	Sec. 2.9.1.1 See Appendix A	
OBJ_VEHICLECLASS	Sec. 2.9.1.1 See Appendix A	
VEC_SMALLER3	Sec. 2.14.3.9 See Appendix A	
OBJ_VELOCITY	Sec. 2.9.1.1 See Appendix A	
VEC_SMALLER2	Sec. 2.14.3.9 See Appendix A	

Table 2.6-300: resupply_check_ok CSU [8.6.6.1.5]

2.6.6.1.6 resupply_bld_ammo_needs CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*array_size	pointer to unsigned char	Standard
array[]	MunitionQuantity	basic.h

Table 2.6-301: resupply_bld_ammo_needs CSU [8.6.6.1.6]

2.6.6.1.7 choose_resupply_item CSU

This CSU builds a list of needs and compares it to a list of items offered, finds an intersection.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
*needs_size	pointer to unsigned char	Standard
needs_ar[]	MunitionQuantity	basic.h
supply_ar[]	MunitionQuantity	basic.h
*munition	pointer to ObjectType	p_sim.h
*quantity	pointer to float	Standard
Calls		
Function	Where Described	
resupply_bld_ammn_needs	Sec. 2.6.6.1.6	

Table 2.6-302: choose_resupply_item CSU [8.6.6.1.7]

2.6.6.1.8 logistics_supply_offer_canceled CSU

This CSU handles the cancel packet when it comes in.

Parameters		
Parameters	Type	Where Typedef Declared
*lgdat	pointer to LOGICTICS_VARS	Sec. 2.9.1.2

Table 2.6-303: logistics_supply_offer_canceled CSU [8.6.6.1.8]

2.6.6.1.9 logistics_supply_offer_received CSU

This CSU handles the packet when it comes in. Array use is as follows:

array[0].quantity : number of array items, including this one
array[0].munition : not used
array[1+].* : munition entries

Parameters		
Parameters	Type	Where Typedef Declared
*lgdat	pointer to LOGICTICS_VARS	Sec. 2.5.1.2
*spdu	pointer to SimulationPDU	p_sim.h
Calls		
Function	Where Described	
saf_id from simnet_id		
DEBUG_VEHICLE	Sec. 2.5.2.2 See Appendix A	
heap_allocate	Sec. 2.14.2.1.1	

Table 2.6-304: logistics_supply_offer_received CSU [8.6.6.1.9]

2.6.6.1.10 logistics_tick CSU

This CSU handles vehicle resupply.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
IS_RESUPPLYING	Sec. 2.6.6.2 See Appendix A	
LOOKUP_SAFOBJ	Sec. 2.9.1.1 See Appendix A	
resupply_check ok	Sec. 2.6.6.1.5	
range_squared	Sec. 2.14.3.5.10	
OBJ_POSITION	Sec. 2.9.1.1 See Appendix A	
DEBUG_VEHICLE	Sec. 2.5.2.2 See Appendix A	
OBJ_VEHICLEID	Sec. 2.9.1.1 See Appendix A	
sbx_printf	Sec. 2.4.3.2.8	
OBJ_OWNER_PORT_NUMBER	Sec. 2.9.1.1 See Appendix A	
choose_resupply_item	Sec. 2.6.6.1.7	
min	Sec. 2.6.7.3 & Sec. 2.13.3.5 See Appendix A	
resupply_bid ammo needs	Sec. 2.6.6.1.6	
simnet_send_resupply_request		
simnet_send_resupply_received		
simnet_send_resupply_cancel		

Table 2.6-305: logistics_tick CSU [8.6.6.1.10]

2.6.6.2 logistics.h CSU

/simnet/src/host/logistics.h

This CSU contains the symbolic constants used by the resupply code, as shown in the following table, and a single macro, IS_RESUPPLYING(a), defined in Appendix A.

Constant	Value
LOGISTICS STATE READY	0
LOGISTICS STATE REQUESTING	1
LOGISTICS STATE RECEIVING	2
RESUPPLY MIN DISTANCE	1000.0
MILLISEC	1000
RESUPPLY STD TIME OUT	40.0 /* sec */
RESUPPLY UNIT AMMO	1.0 /* rounds per RESUPPLY TIME UNIT */
RESUPPLY UNIT FUEL	20.0 /* gallons per RESUPPLY TIME UNIT */
RESUPPLY UNIT FUEL SEC	1600.0 /* seconds of fuel per " */
RESUPPLY FUEL SEC PER SEC	40
RESUPPLY FUEL GAL PER SEC	.5
RESUPPLY FUEL GAL TO SEC	80

Table 2.6-306: logistics.h Constant Definitions

2.6.7 libpvis CSC

/simnet/libsrc/libpvis

This library deals with issues concerning the intervisibility; the total, partial, or lack of visibility of objects by the eyes of the observer.

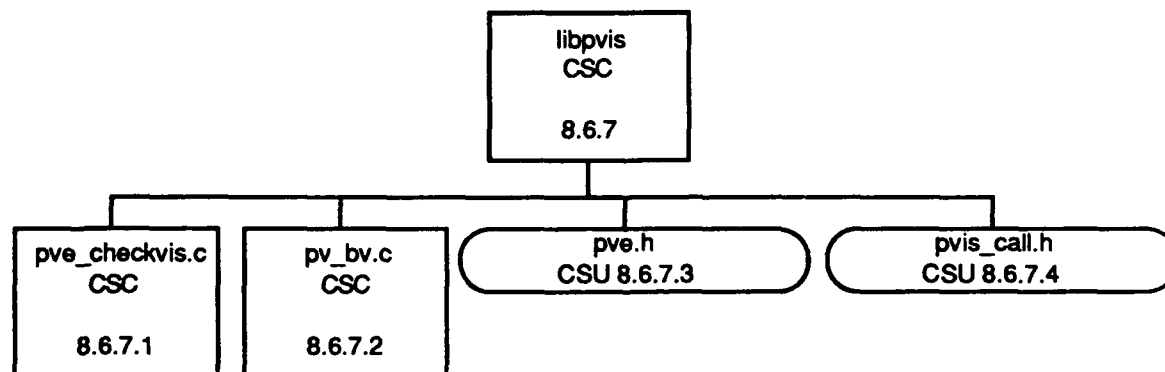


Figure 2.6-23: pve_checkvis.c CSC Structure

There are a number of factors (conditional definitions) that determine what (and how) this library does in terms of type of calculations, database format, debugging, etc. The setting of constant defines is done in both pve_checkvis.c (see Sec. 2.6.7.1) and pvis_call.h (see Sec. 2.6.7.4). A define of 0 means that the code segment indicated is undefined and not to be compiled and executed, while a value of 1 causes the code to be included in the compile.

2.6.7.1 pve_checkvis.c CSC

/simnet/libsrc/libpvis/pve_checkvis.c

This CSC contains CSUs which determine the degree of visibility of objects.

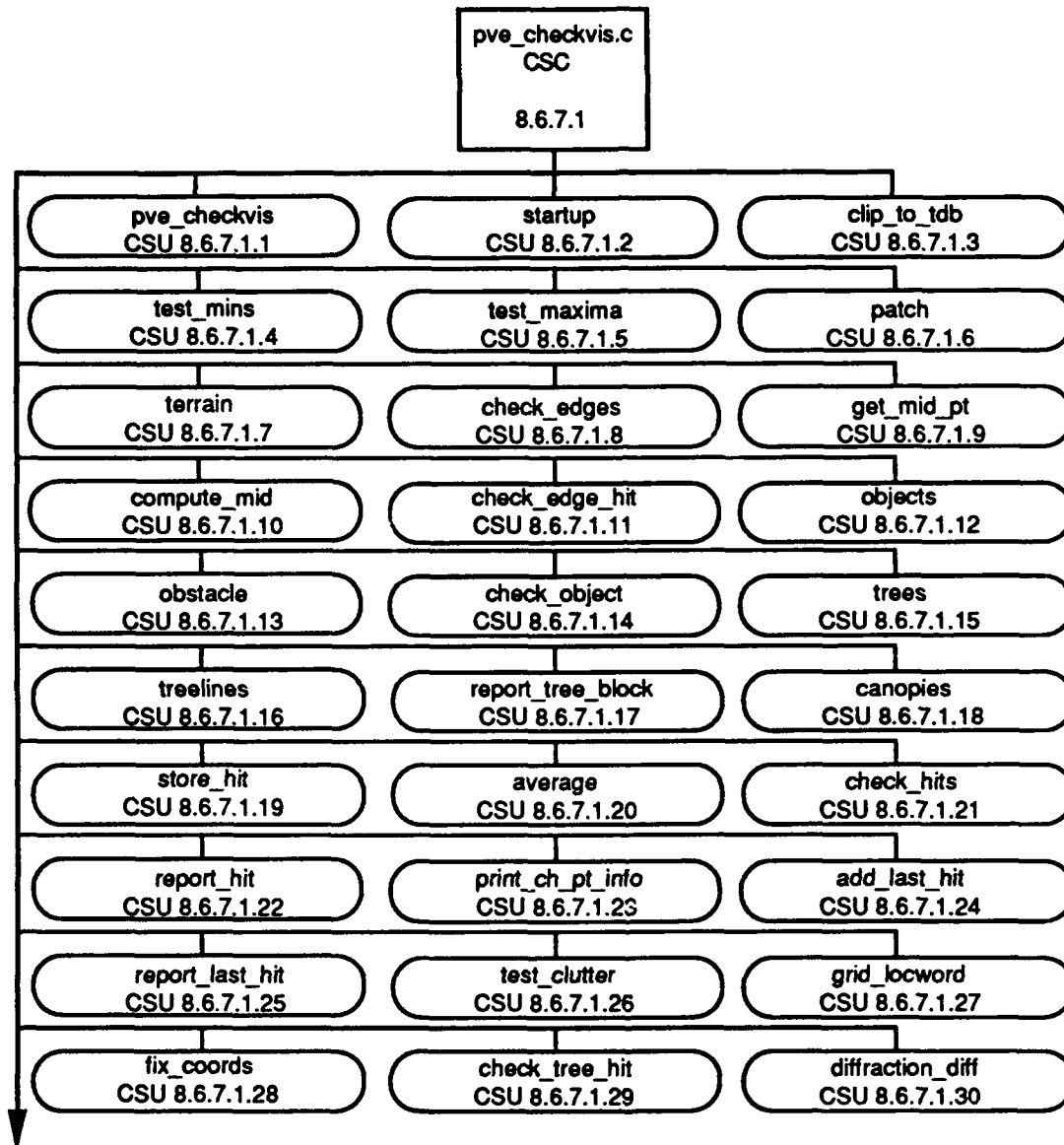


Figure 2.6-24: pve_checkvis.c CSC Structure Part 1 of 2

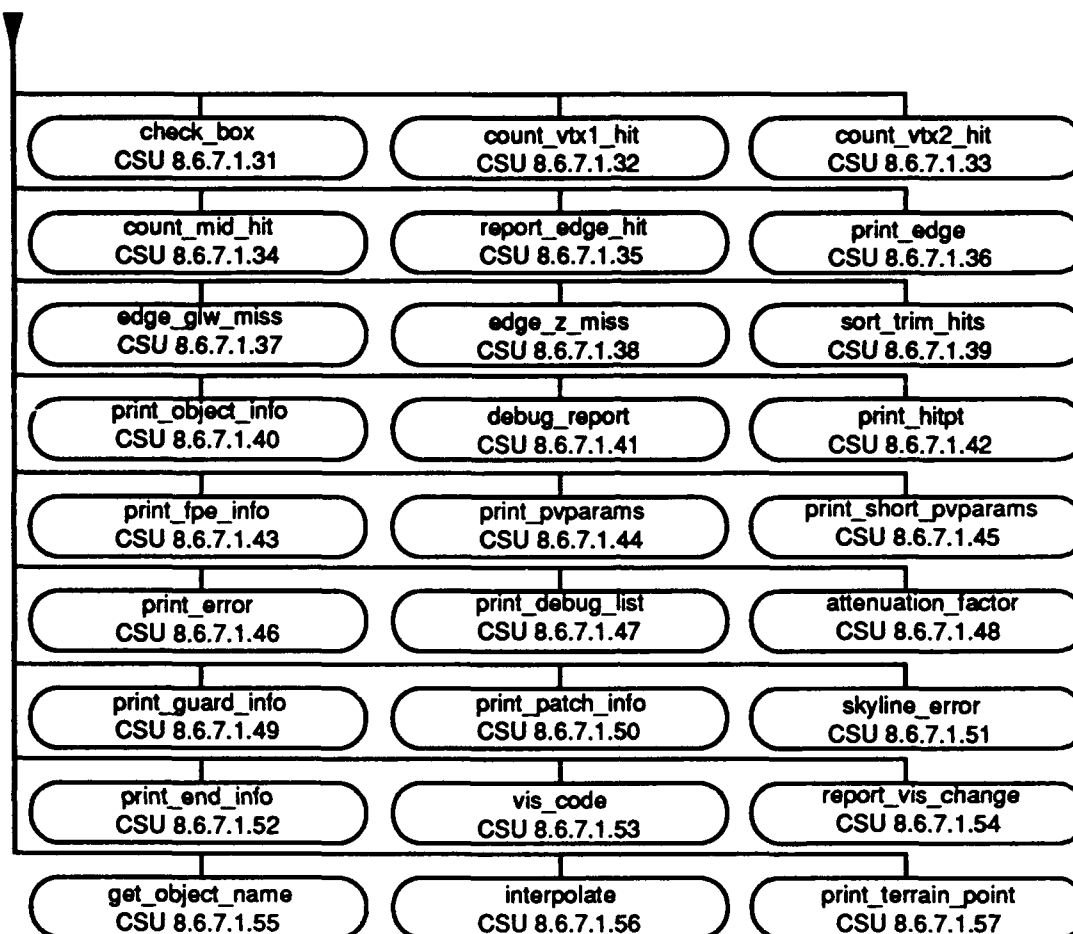


Figure 2.6-25: pve_checkvis.c CSC Structure Part 2 of 2

The following table contains the constant defines in pve_checkvis.c that affect compilation, and subsequent library action. With them are their current values and a brief explanation of their meanings.

Constant	Value
ALLOW_SEG_VIS	1 /* Allow the use of segmented intervis math */
REPORT_BOTH_EDGE_BLOCKS	1 /* Report both horizontal and vertical grid line blockage in test mins() */
TDB_2	1 /* Use new database format for Bergen.0102 and Knox .030x*/
NEW_DIST	1 /* Use new distance methods in checking edges */
IGNORE_LOW_TREELINES	1 /* Treelines below MIN_TREELINE_HEIGHT are ignored */
PRINT_OBJECTS	0 /* Objects are not printed */
PRINT_ERRORS	0 /* Debugging errors are not printed */
QUIT_ON_ERRORS	0 /* Debugging errors don't cause terminations */
DEBUG_PATCHES	0 /* The following are the various debug code */
DEBUG_LIST_EDGES	0 /* defines. Since all are OFF (undefined) none */
DEBUG_EDGES	0 /* of debug trapping code is compiled. In */
DEBUG_REPORT	0 /* addition, the constant DEBUG is defined */
DEBUG_TREES	0 /* below as 0, since it is the product of the logical */
DEBUG_CANOPIES	0 /* OR of the other smaller debug defines. */
DEBUG_HITS	0
DEBUG_HITS_2	0
DEBUG_TREE_OBJ_ELEVS	0
DEBUG_ELEV	0
DEBUG_MAX_ELEV	0
DEBUG_ADD	0
DEBUG_OBJECTS	0
DEBUG_CLUTTER	0
DEBUG_SWITCHES	0
DEBUG_CLIP	0
DEBUG_CHECK_BOX	0
DEBUG_DIFFRACTION	0
DEBUG_OBSTACLES	0
DEBUG	(DEBUG_PATCHES DEBUG_LIST_EDGES DEBUG_EDGES DEBUG_TREES DEBUG_CLUTTER DEBUG_OBJECTS DEBUG_CANOPIES DEBUG_SWITCHES DEBUG_HITS DEBUG_CLIP DEBUG_REPORT DEBUG_CHECK_BOX DEBUG_DIFFRACTION DEBUG_OBSTACLES DEBUG_TREE_OBJ_ELEVS DEBUG_ELEV DEBUG_MAX_ELEV DEBUG_ADD)

Table 2.6-307: pve_checkvis.c Test and Debug Constant Definitions

In addition to the constant definitions there is one for the diffraction boundary weight in the following table and values for the ql_tbl array. (See pve_checkvis.c for the contents of the array.)

Constant	Value
DIFFRAC_BDRY_WT	100

Table 2.6-308: DIFFRAC_BDRY_WT Constant Definition

2.6.7.1.1 pve_checkvis CSU

This CSU is the main routine for checking the invisibility of objects.

Parameters		
Parameters	Type	Where Typedef Declared
*pvparams	pointer to PV_PARAMS	Sec. 2.6.7.4
ReturnValues		
Return Value	Type	Meaning
DUMMY	int	Abnormal end.
FULLY_VISIBLE	int	Object is fully visible.
INVISIBLE	int	Object is invisible.
visibility	int	Visibility level.
Errors		
Error Name	Reason for Error	
"error opening file pve_pt_stats"	Unable to open file pve_pt_stats.	
"error opening file pve_stats"	Unable to open file pve_stats.	
"NULL pvparams passed to pve_checkvis. Bye.."	Null pv parameters were passed to pve_checkvis.	
"NULL pointer guardp passed to pve_checkvis. Bye.."	Null pointer guardp was passed to pve_checkvis.	
Calls		
Function	Where Described	
print_pvparams	Sec. 2.6.7.1.44	
startup	Sec. 2.6.7.1.2	
test_mins	Sec. 2.6.7.1.4	
diffraction_diff	Sec. 2.6.7.1.30	
test_maxima	Sec. 2.6.7.1.5	
report_last_hit	Sec. 2.6.7.1.25	
test_clutter	Sec. 2.6.7.1.26	
print_end_info	Sec. 2.6.7.1.52	

Table 2.6-309: pve_checkvis CSU [8.6.7.1.1]

2.6.7.1.2 startup CSU

This CSU sets up the geometry for traversing the terrain patches. It begins point computations where the ray from eye to target hits the patch boundaries. It terminates the CSU call if the eye is too close to the target.

ReturnValues		
Return Value	Type	Meaning
TERMINATE	static int	Abnormal end.
DUMMY	static int	Normal end.
Errors		
Error Name	Reason for Error	
"unable to open file pve_errors"	Unable to open file pve_errors.	
Calls		
Function	Where Described	
max	Sec. 2.6.7.3 & Sec. 2.13.3.5 See Appendix A	
clip to tdb	Sec. 2.6.7.1.3	
PATCH ROUND UP	Sec. 2.6.7.3 See Appendix A	
min	Sec. 2.6.7.3 & Sec. 2.13.3.5 See Appendix A	
PATCH ROUND DOWN	Sec. 2.6.7.3 See Appendix A	
DIST	Sec. 2.6.7.3 See Appendix A	

Table 2.6-310: startup CSU [8.6.7.1.2]

2.6.7.1.3 clip_to_tdb CSU

Parameters		
Parameters	Type	Where Typedef Declared
x_1	double	Standard
y_1	double	Standard
x_2	double	Standard
y_2	double	Standard
ReturnValues		
Return Value	Type	Meaning
TERMINATE	static int	Abnormal end.
DUMMY	static int	Normal end
Errors		
Error Name	Reason for Error	
"tdb_get_z error at next.x, next.y"	A call to the function tdb_get_z returned -1.	

Calls	
Function	Where Described
DIST	Sec. 2.6.7.3 See Appendix A
tdb_get_z	Sec. 2.21.7.16.2
print_error	Sec. 2.6.7.1.46
max	Sec. 2.6.7.3 & Sec. 2.13.3.5 See Appendix A

Table 2.6-311: clip_to_tdb CSU [8.6.7.1.3]

2.6.7.1.4 test_mins CSU

This CSU checks the elevation on the patch boundaries.

ReturnValues		
Return Value	Type	Meaning
BLOCKED	static int	The minimum elevation on the boundary is above the ray from the eye to the target top (so the target will be invisible).
DUMMY	static int	Any other situation.
Calls		
Function	Where Described	
DIST	Sec. 2.6.7.3 See Appendix A	
PATCH_INDEX		
print_guard_info	Sec. 2.6.7.1.49	
max	Sec. 2.6.7.3 & Sec. 2.13.3.5 See Appendix A	

Table 2.6-312: test_mins CSU [8.6.7.1.4]

2.6.7.1.5 test_maxima CSU

This CSU computes the points where the ray hits successive patch boundaries. It also computes the minimum elevation of the ray in the patch, and checks this against the maximum elevation of terrain polygons, bounding volumes, and trees.

ReturnValues		
Return Value	Type	Meaning
height_pt	static int	the height point
Errors		
Error Name	Reason for Error	
"tdb_get_terrain error"	A call to function tdb_get_terrain returned NULL.	
"tdb_get_terrain on ... moving on to next patch...."	A call to function tdb_get_terrain returned NULL.	

Calls	
Function	Where Described
DIST	Sec. 2.6.7.3 See Appendix A
min	Sec. 2.6.7.3 & Sec. 2.13.3.5 See Appendix A
PATCH INDEX	
print guard info	Sec. 2.6.7.1.49
max	Sec. 2.6.7.3 & Sec. 2.13.3.5 See Appendix A
fix coords	Sec. 2.6.7.1.28
SCALE COORDS	Sec. 2.6.7.3
grid locword	Sec. 2.6.7.1.27
tdb get terrain	Sec. 2.21.7.18.1
print error	Sec. 2.6.7.1.46
tdb error	Sec. 2.21.7.17.1
patch	Sec. 2.6.7.1.6

Table 2.6-313: test_maxima CSU [8.6.7.1.5]

2.6.7.1.6 patch CSU

This CSU checks all visibility in a terrain patch.

Errors	
Error Name	Reason for Error
"results overflow, not enough memory allocated for PV_CHANGE_PTs for pve_checkvis Abandoning computation of segmented intervisibility"	A call to check_hits returned RES_OFLOW.
Calls	
Function	Where Described
print_patch info	Sec. 2.6.7.1.50
terrain	Sec. 2.6.7.1.7
objects	Sec. 2.6.7.1.12
trees	Sec. 2.6.7.1.15
treelines	Sec. 2.6.7.1.16
canopies	Sec. 2.6.7.1.18
sort trim hits	Sec. 2.6.7.1.39
check hits	Sec. 2.6.7.1.21

Table 2.6-314: patch CSU [8.6.7.1.6]

2.6.7.1.7 terrain CSU

This CSU checks the terrain edge list for the patch, testing edges matching grid_loc_word for intervisibility blocks.

Calls	
Function	Where Described
check edges	Sec. 2.6.7.1.8

Table 2.6-315: terrain CSU [8.6.7.1.7]

2.6.7.1.8 check_edges CSU

This CSU checks an edge list for the patch, testing edges, matching grid_loc_word for intervisibility blocks.

Parameters		
Parameters	Type	Where Typedef Declared
edge	register int	Standard
*verticesp	pointer to register TERRAIN_POINT	terrain.h
*edgesp	pointer to register EDGE_DESCRIPTOR	terrain.h
type	register int	Standard
Calls		
Function	Where Described	
print edge	Sec. 2.6.7.1.36	
edge_glw_miss	Sec. 2.6.7.1.37	
edge_z_miss	Sec. 2.6.7.1.38	
POINT TO LINE	Sec. 2.6.7.3 See Appendix A	
report edge hit	Sec. 2.6.7.1.35	
count vtx1 hit	Sec. 2.6.7.1.32	
check edge hit	Sec. 2.6.7.1.11	
DIST	Sec. 2.6.7.3 See Appendix A	
count vtx2 hit	Sec. 2.6.7.1.33	
count mid hit	Sec. 2.6.7.1.34	
get mid pt	Sec. 2.6.7.1.9	
compute mid	Sec. 2.6.7.1.10	

Table 2.6-316: check_edges CSU [8.6.7.1.8]

2.6.7.1.9 get_mid_pt CSU

Parameters		
Parameters	Type	Where Typedef Declared
*newptp	pointer to TERRAIN_POINT	terrain.h
*pt1p	pointer to TERRAIN_POINT	terrain.h
*pt2p	pointer to TERRAIN_POINT	terrain.h
lambd _a	register double	Standard

Table 2.6-317: get_mid_pt CSU [8.6.7.1.9]

2.6.7.1.10 compute_mid CSU

Parameters		
Parameters	Type	Where Typedef Declared
*midpointp	pointer to register TERRAIN_POINT	terrain.h
*point1p	pointer to register TERRAIN_POINT	terrain.h
weight1	register double	Standard
*point2p	pointer to register TERRAIN_POINT	terrain.h

Table 2.6-318: compute_mid CSU [8.6.7.1.10]

2.6.7.1.11 check_edge_hit CSU

Parameters		
Parameters	Type	Where Typedef Declared
*pointp	pointer to register TERRAIN_POINT	terrain.h
type	register int	Standard
ReturnValues		
Return Value	Type	Meaning
TERMINATE	int	Checking point-to-point only and visibility is zero.
DUMMY	int	Normal end.
Calls		
Function	Where Described	
DIST	Sec. 2.6.7.3 See Appendix A	
print terrain point	Sec. 2.6.7.1.57	
report_vis_change	Sec. 2.6.7.1.54	
max	Sec. 2.6.7.3 & Sec. 2.13.3.5 See Appendix A	
store_hit	Sec. 2.6.7.1.19	

Table 2.6-319: check_edge_hit CSU [8.6.7.1.11]

2.6.7.1.12 objects CSU

This CSU checks the bounding volumes.

Calls	
Function	Where Described
GET OBJECT MIN X	terrain.h
GET OBJECT MIN Y	terrain.h
GET OBJECT MAX X	terrain.h
GET OBJECT MAX Y	terrain.h
GET OBJECT MAX Z	terrain.h
GET OBJECT TYPE	terrain.h
check_box	Sec. 2.6.7.1.31
obstacle	Sec. 2.6.7.1.13
check_object	Sec. 2.6.7.1.14

Table 2.6-320: objects CSU [8.6.7.1.12]

2.6.7.1.13 obstacle CSU

Calls	
Function	Where Described
DIST	Sec. 2.6.7.3
min	Sec. 2.6.7.3 & Sec. 2.13.3.5 See Appendix A

Table 2.6-321: obstacle CSU [8.6.7.1.13]

2.6.7.1.14 check_object CSU

This CSU tests the given object for a visibility blockage.

Parameters		
Parameters	Type	Where Typedef Declared
*objectp	pointer to OBJECT_DESCRIPTOR	terrain.h
Calls		
Function	Where Described	
print_object_info	Sec. 2.6.7.1.40	
DIST	Sec. 2.6.7.3	
POINT TO LINE	Sec. 2.6.7.3 See Appendix A	
max	Sec. 2.6.7.3 & Sec. 2.13.3.5 See Appendix A	
store_hit	Sec. 2.6.7.1.19	

Table 2.6-322: check_object CSU [8.6.7.1.14]

2.6.7.1.15 trees CSU

This CSU checks trees. A tree is modeled as a triangle T_LINE_BOT_WINDOW above the ground. z_hit is not quite correct as ground height, but it is used anyway. This is not a problem for point-to-point intervisibility. z_hit is correct if the ground is flat or dot1 is 0.

Calls	
Function	Where Described
DIST	Sec. 2.6.7.3
check tree hit	Sec. 2.6.7.1.29
abs	Sec. 2.6.7.3 & Sec. 2.13.3.2 See Appendix A

Table 2.6-323: trees CSU [8.6.7.1.15]**2.6.7.1.16 treelines CSU**

If the ray goes through the guard box around the treeline, this CSU checks for blockage at the tree at each end of the treeline, and for blockage in the segments. It terminates if there are one or fewer segments.

Parameters		
Parameters	Type	Where Typedef Declared
num tlines	register int	Standard
*treelinep	pointer to register TREELINE HEADER	terrain.h

Errors	
Error Name	Reason for Error
"treeline error. treeline"	There are one or fewer treeline vertices.
"treeline error: ... vertices from ... to ... last: ..., next: ..."	There are one or fewer treeline vertices.

Calls	
Function	Where Described
check box	Sec. 2.6.7.1.31
print error	Sec. 2.6.7.1.46
POINT TO LINE	Sec. 2.6.7.3 See Appendix A
DIST	Sec. 2.6.7.3 See Appendix A
check tree hit	Sec. 2.6.7.1.29
compute mid	Sec. 2.6.7.1.10

Table 2.6-324: treelines CSU [8.6.7.1.16]

2.6.7.1.17 report_tree_block CSU

This CSU puts a tree hit in the PV_RESULT structure. It is called only if params->num_trees < PV_MAX_TREE_BLOCKS (which means that more tree blocks may be reported) or if this tree is closer than the furthest one previously reported.

Parameters		
Parameters	Type	Where Typedef Declared
x	REAL 4	mass stdc.h
y	REAL 4	mass stdc.h
dist	double	Standard
Calls		
Function	Where Described	
min	Sec. 2.6.7.3 & Sec. 2.13.3.5 See Appendix A	
tree_dist		

Table 2.6-325: report_tree_block CSU [8.6.7.1.17]

2.6.7.1.18 canopies CSU

This CSU checks the canopy edge list for the patch, testing edges matching grid_loc_word for intervisibility blocks.

Calls	
Function	Where Described
check_box	Sec. 2.6.7.1.31
treelines	Sec. 2.6.7.1.16
check_edges	Sec. 2.6.7.1.8

Table 2.6-326: canopies CSU [8.6.7.1.18]

2.6.7.1.19 store_hit CSU

This CSU stores a hit point as follows: For objects, z represents the elevation of the top with z_b_in and z_b_out representing surface-level elevations relative to z_from. For trees, z_b_in is surface elevation, z is top elevation, z_b_out is the elevation of the bottom of the foliage, T_LINE_BOT_WINDOW above the ground. For tree canopies and terrain edges, though, z equals z_b_in equals z_b_out. In particular, for canopy edges the ground elevation is unknown.

Parameters		
Parameters	Type	Where Typedef Declared
x	double	Standard
y	double	Standard
z	double	Standard
z_b_in	double	Standard
z_b_out	double	Standard
type	int	Standard
d	double	Standard
ratio	double	Standard

Errors	
Error Name	Reason for Error
"OVERFLOW, ... hits in patch, at ... Only ... hits may be recorded in a patch"	Total hits exceeded maximum hit points.

Table 2.6-327: store_hit CSU [8.6.7.1.19]

2.6.7.1.20 average CSU

This CSU computes the linearly weighted average of elevations z1 at distance d1 and z2 at distance d2, at an intermediate distance d3.

Parameters		
Parameters	Type	Where Typedef Declared
z1	register double	Standard
d1	register double	Standard
z2	register double	Standard
d2	register double	Standard
d3	register double	Standard
ReturnValues		
Return Value	Type	Meaning
$(z1 + z2) / 2$	static double	d1 and d2 are about equal.
$((d3 - d1) / (d2 - d1)) * z2 + ((d2 - d3) / (d2 - d1)) * z1$	static double	Linearly weighted average.

Table 2.6-328: average CSU [8.6.7.1.20]

2.6.7.1.21 check_hits CSU

This CSU checks the hits for a given patch, reporting any changes in visibility. No distinction is made between full and partial tree blocks. All tree blocks are called partial.

ReturnValues		
Return Value	Type	Meaning
RES_OFLOW	static int	Change points >= (maximum change points minus 2).
DUMMY	static int	Normal end.
Errors		
Error Name	Reason for Error	
"pve_checkvis: check_hits switch error"	Value in curp->type is none of the expected types.	
"pve_checkvis: check_hits switch error curp ..., type ..., dist ..."	Value in curp->type is none of the expected types.	

Calls	
Function	Where Described
vis code	Sec. 2.6.7.1.53
report hit	Sec. 2.6.7.1.22
max	Sec. 2.6.7.3 & Sec. 2.13.3.5 See Appendix A

Table 2.6-329: check_hits CSU [8.6.7.1.21]

2.6.7.1.22 report_hit CSU

ReturnValues		
Return Value	Type	Meaning
TERMINATE	static int	Any of various distance variables is greater than total distance.
DUMMY	static int	Standard
Errors		
Error Name	Reason for Error	
**** skyline error in report_hit	Case is INVISIBLE and skyline_denom <= 0.	
"skyline error"	Case is INVISIBLE, and skyline_denom <= 0; or case is FULL_BLK_TREES; or case is PT_BLK_SOLID_PT_BLK_TREES and skyline_denom <= 0; or case is PT_BLK_TREES and skyline_denom >= 0; or case is PT_BLK_SOLID and do_trees = TRUE; or case is PT_BLK_SOLID and inv_part_solid_dist > currp->dist; or case is PT_BLK_SOLID and vis_part_solid_dist > currp->dist; or case is PT_BLK_SOLID and not(skyline_denom < 0 && currvs == INVISIBLE) and not (skyline_denom > 0 && currvs == FULLY_VISIBLE); or case is FULLY_VISIBLE and skyline_denom >= 0.	
"case error, report_hit"	The value in old_vis is PT BLK SOLID FULL BLK TREES.	

Table 2.6-330 is continued on the following page

"pve_checkvis: skyline error in report_hit"	Case is PT_BLK_SOLID_FULL_BLK_TREES; or case is PT_BLK_SOLID_PT_BLK_TREES and skyline_denom <= 0; or case is PT_BLK_TREES and skyline_denom >= 0; or case is FULLY_VISIBLE and skyline_denom >= 0.
"pve_checkvis: skyline error in report_hit report_hit error: old_vis ... case not implemented"	Case is FULL_BLK_TREES. .
"pve_checkvis: treeline error in report_hit"	Case is PT_BLK_TREES and treeline_denom <= 0.
"treeline error"	Case is PT_BLK_TREES and treeline_denom <= 0.
"report_hits error"	Case is PT_BLK_TREES and vis_part_tree_dist > currp->dist.
**** ERROR in report_hits ****	Case is PT_BLK_TREES and vis_part_tree_dist > currp->dist.
"pve_checkvis: report_hit error: PART_BLK_SOLID & do_trees"	Case is PT_BLK_SOLID and do_trees = TRUE.
"pve_checkvis: skyline error in report_hit, old_vis PT_BLK_SOLID"	Case is PT_BLK_SOLID and inv_part_solid_dist > currp->dist; or case is PT_BLK_SOLID and vis_part_solid_dist > currp->dist; or case is PT_BLK_SOLID and not(skyline_denom < 0 && currv == INVISIBLE) and not (skyline_denom > 0 && currv == FULLY_VISIBLE).
"pve_checkvis: unknown case in report_hit"	Value of old_vis is none of the expected visibilities.
"report_hit error: unknown case: hits ..."	Value of old_vis is none of the expected visibilities.
"unknown case in report_hit"	Value of old_vis is none of the expected visibilities.
Calls	
Function	Where Described
print hitpt	Sec. 2.6.7.1.42
debug report	Sec. 2.6.7.1.41
print ch_pt info	Sec. 2.6.7.1.23
print error	Sec. 2.6.7.1.46
skyline_error	Sec. 2.6.7.1.51
XC	Sec. 2.6.7.3 See Appendix A
YC	Sec. 2.6.7.3 See Appendix A
vis_code	Sec. 2.2.6.7.1.53

Table 2.6-330: report_hit CSU [8.6.7.1.22]

2.6.7.1.23 print_ch_pt_info CSU

This CSU writes change-point information to a file.

Parameters		
Parameters	Type	Where Typedef Declared
*fp	pointer to FILE	Standard
ch_dist	double	Standard
ch_vis	int	Standard
Calls		
Function	Where Described	
XC	Sec. 2.6.7.3 See Appendix A	
YC	Sec. 2.6.7.3 See Appendix A	
vis_code	Sec. 2.6.7.1.53	

Table 2.6-331: print_ch_pt_info CSU [8.6.7.1.23]

2.6.7.1.24 add_last_hit CSU

Errors	
Error Name	Reason for Error
"add_last_hit error"	curp == NULL
Calls	
Function	Where Described
print_error	Sec. 2.6.7.1.46
print_hitpt	Sec. 2.6.7.1.42

Table 2.6-332: add_last_hit CSU [8.6.7.1.24]

2.6.7.1.25 report_last_hit CSU

Errors	
Error Name	Reason for Error
"segvis != ptvis error"	old_vis and possibly see under tree have improper values.
Calls	
Function	Where Described
print_error	Sec. 2.6.7.1.46
interpolate	Sec. 2.6.7.1.56
DIST	Sec. 2.6.7.3 See Appendix A

Table 2.6-333: report_last_hit CSU [8.6.7.1.25]

2.6.7.1.26 test_clutter CSU

This CSU checks from target position to the edge of the tdb to see if a terrain patch behind the target is above the line of sight from eye to the top of the target.

Calls	
Function	Where Described
DIST	Sec. 2.6.7.3 See Appendix A
STILL IN BOUNDS X	Sec. 2.6.7.3 See Appendix A
STILL IN BOUNDS Y	Sec. 2.6.7.3 See Appendix A
PATCH INDEX	tdb in MCC CSCI SDD
print guard info	Sec. 2.6.7.1.49

Table 2.6-334: test_clutter CSU [8.6.7.1.26]

2.6.7.1.27 grid_locword CSU

This CSU looks up the grid locator word of the ray between a point on *edge 1* with coordinate *coord 1*, to *edge 2*, *coord 2*. Edges are called PVE_LEFT, PVE_TOP, PVE_RIGHT, and PVE_BOTTOM. Coordinates are measured from the bottom or left, and are 0, 1, 2, ... (PV_REF_GR_NUM - 1). Caller is responsible for ensuring that these coordinates are integers in this range.

PV_REF_GR_NUM == REFINEMENT_LEVEL * PV_GRID_NUM
 PV_GRID_NUM is the number of squares along an edge of the grid that is used in the definition of the grid locator word.

Parameters		
Parameters	Type	Where Typedef Declared
edge_1	int	Standard
coord_1	int	Standard
edge_2	int	Standard
coord_2	int	Standard
Return Values		
Return Value	Type	Meaning
PVE_GLW_LEFT	static unsigned	Case edge_1, PVE_LEFT, edge 2, PVE LEFT.
gl_tbl[]	static unsigned	All other known cases.
PVE_GLW_ERROR	static unsigned	Unknown cases.
PVE_GLW_TOP	static unsigned	Case edge_1, PVE_TOP, edge 2, PVE TOP.
PVE_GLW_RIGHT	static unsigned	Case edge_1, PVE_RIGHT, edge 2, PVE RIGHT.
PVE_GLW_BOTTOM	static unsigned	Case edge_1, PVE_BOTTOM, edge_2, PVE BOTTOM.

Table 2.6-335: grid_locword CSU [8.6.7.1.27]

2.6.7.1.28 fix_coords CSU

This CSU computes `last_coord_dbl` and `next_coord_dbl` (patch coordinates). They are doubles between 0 and `patch_size`. `last_coord_dbl` is modified slightly to ensure that it will represent patch coordinates in the patch of `mid`.

Calls	
Function	Where Described
PATCH_COORD	Sec. 2.6.7.3 See Appendix A

Table 2.6-336: fix_coords CSU [8.6.7.1.28]

2.6.7.1.29 check_tree_hit CSU

For point-to-point intervisibility, this CSU reports a partial tree block if:

- the bottom of the tree is below the top of the target and
- the top of the tree is above the bottom of the target;

reports a full tree block if:

- the bottom of the tree is below the bottom of the target and
- the top of the tree is above the top of the target.

It does not check cumulative effects of more than one tree, which might give a full tree block by a combination of partial blocks. All elevations are relative to `z_from`.

Parameters		
Parameters	Type	Where Typedef Declared
x	double	Standard
y	double	Standard
dist	register double	Standard
tree_top	register double	Standard
tree_bot	register double	Standard
ground_elev	register double	Standard
type	int	Standard
Calls		
Function	Where Described	
max	Sec. 2.6.7.3 & Sec. 2.13.3.5 See Appendix A	
store_hit	Sec. 2.6.7.1.19	
report_tree_block	Sec. 2.6.7.1.17	

Table 2.6-337: check_tree_hit CSU [8.6.7.1.29]

2.6.7.1.30 diffraction_diff CSU

Parameters		
Parameters	Type	Where Typedef Declared
forward_ratio	double	Standard
reverse_ratio	double	Standard

ReturnValues		
Return Value	Type	Meaning
0.0	static double	forward_ratio <= top_ratio or numer_ground <= 0.
dist	static double	Normal end.
Calls		
Function	Where Described	
attenuation factor	Sec. 2.6.7.1.48	

Table 2.6-338: diffraction_diff CSU [8.6.7.1.30]

2.6.7.1.31 check_box CSU

This CSU returns a if a line through (x_last, y_last) perpendicular to the vector (x_normal, y_normal) intersects the box with corners (x_min, y_min), (x_max, y_min), (x_max, y_max), (x_min, y_max). For floating point error reasons, it tests for product < EPSI_EPSI rather than product < 0.

Parameters		
Parameters	Type	Where Typedef Declared
x min	register REAL 4	mass stdc.h
y min	register REAL 4	mass stdc.h
x max	register REAL 4	mass stdc.h
y max	register REAL 4	mass stdc.h
ReturnValues		
Return Value	Type	Meaning
1	int	Line intersects box.
0	int	Line does not intersect box.

Table 2.6-339: check_box CSU [8.6.7.1.31]

2.6.7.1.32 count_vtx1_hit CSU

This CSU increments the count of terrain or canopy vtx1-hits.

Parameters		
Parameters	Type	Where Typedef Declared
type	int	Standard

Table 2.6-340: count_vtx1_hit CSU [?????]

2.6.7.1.33 count_vtx2_hit CSU

This CSU increments the count of terrain or canopy vtx2-hits.

Parameters		
Parameters	Type	Where Typedef Declared
type	int	Standard

Table 2.6-341: count_vtx2_hit CSU [?????]

2.6.7.1.34 count_mid_hit CSU

This CSU increments the count of terrain or canopy mid-hits.

Parameters		
Parameters	Type	Where Typedef Declared
type	int	Standard

Table 2.6-342: count_mid_hit CSU [?????]

2.6.7.1.35 report_edge_hit CSU

This CSU writes to a file: message, edge, dot, and the x, y, z, and (z - z_from) for the vertex point for an edge hit.

Parameters		
Parameters	Type	Where Typedef Declared
*msg	pointer to char	Standard
edge	int	Standard
*vertp	pointer to TERRAIN_POINT	terrain.h
dot	double	Standard

Table 2.6-343: report_edge_hit CSU [?????]

2.6.7.1.36 print_edge CSU

This CSU writes to a file: x, y, z, and (z - z_from) for the two vertices forming an edge.

Parameters		
Parameters	Type	Where Typedef Declared
edge	int	Standard
*vertex1p	pointer to TERRAIN_POINT	terrain.h
*vertex2p	pointer to TERRAIN_POINT	terrain.h

Table 2.6-344: print_edge CSU [?????]

2.6.7.1.37 edge_glw_miss CSU

This CSU writes to a file the grid loc word miss for edge, edge_glw, and ray_glw.

Parameters		
Parameters	Type	Where Typedef Declared
edge	int	Standard
edge_glw	HWORD	mass_std.h
ray_glw	HWORD	mass_std.h

Table 2.6-345: edge_glw_miss CSU [?????]

2.6.7.1.38 edge_z_miss CSU

This CSU writes to a file the elevation miss of an edge, and z_ray_min.

Parameters		
Parameters	Type	Where Typedef Declared
edge	int	Standard

Table 2.6-346: edge_z_miss CSU [?????]

2.6.7.1.39 sort_trim_hits CSU

Errors	
Error Name	Reason for Error
"sort error"	(currp->dist + EPSI_EPSI) < header_hitp->nextp->dist
"sort error: curr dist ... firstdist ... currp ..., header_hitp->nextp ..."	(currp->dist + EPSI_EPSI) < header_hitp->nextp->dist
"terr_nextp case error"	Value in currp->type is none of the expected types.
"terr_nextp case ... error in standard branch: surp ..., terr_nextp ... currp dist ..., terr_nextp dist ..."	Value in currp->type is none of the expected types.
"pve_checkvis: terr_nextp case error"	Value in currp->type is none of the expected types.
"tdb_get z error at ..."	A call to function tdb_get z returned -1.
"tdb_get z error"	A call to function tdb_get z returned -1.
Calls	
Function	Where Described
print_debug_list	Sec. 2.6.7.1.47
print_error	Sec. 2.6.7.1.46
average	Sec. 2.6.7.1.20
tdb_get_z	Sec. 2.21.7.16.2
max	Sec. 2.6.7.3 & Sec. 2.13.3.5 See Appendix A
add_last_hit	Sec. 2.6.7.1.24

Table 2.6-347: sort_trim_hits CSU [?????]

2.6.7.1.40 print_object_info CSU

This CSU writes to a file the name, height, type, and vertices of an object.

Parameters		
Parameters	Type	Where Typedef Declared
*fp	pointer to FILE	Standard
*objp	pointer to OBJECT_DESCRIPTOR	terrain.h
Calls		
Function	Where Described	
get_object_name	Sec. 2.6.7.1.55	
GET_OBJECT_TYPE	terrain.h	

Table 2.6-348: print_object_info CSU [?????]

2.6.7.1.41 debug_report CSU

This CSU writes to a file debug information as directed by the argument *n*. For *n*=1, distance, visibility, and skyline-ratios are written. For *n*=2, variables of the types *_skyline*, *numer_*, and *_denom* are written.

Parameters		
Parameters	Type	Where Typedef Declared
<i>n</i>	int	Standard

Table 2.6-349: debug_report CSU [?????]

2.6.7.1.42 print_hitpt CSU

This CSU writes to a file hit point information from a PV_HIT_PT data structure.

Parameters		
Parameters	Type	Where Typedef Declared
*pointp	pointer to PV_HIT_PT	Sec. 2.6.7.4

Table 2.6-350: print_hitpt CSU [?????]

2.6.7.1.43 print_fpe_info CSU

This CSU writes to a file fpe error information.

Parameters		
Parameters	Type	Where Typedef Declared
vis1	int	Standard
vis2	int	Standard

Errors	
Error Name	Reason for Error
"error opening file pve_stats"	Unable to open file pve_stats to write.

Table 2.6-351: print_fpe_info CSU [?????]

2.6.7.1.44 print_pvparams CSU

This CSU writes to a file information from a PV_PARAMS data structure.

Parameters		
Parameters	Type	Where Typedef Declared
*file	pointer to FILE	Standard
*params	pointer to PV_PARAMS	Sec. 2.6.7.4

Table 2.6-352: print_pvparams CSU [?????]

2.6.7.1.45 print_short_pvparams CSU

This CSU writes to a file only the visibility test information from a PV_PARAMS data structure.

Parameters		
Parameters	Type	Where Typedef Declared
*file	pointer to FILE	Standard
*params	pointer to PV_PARAMS	Sec. 2.6.7.4

Table 2.6-353: print_short_pvparams CSU [?????]

2.6.7.1.46 print_error CSU

This CSU writes to a file an error message with distances and visibilities and then calls print_short_pvparams to write visibility test information

Parameters		
Parameters	Type	Where Typedef Declared
*fp	pointer to FILE	Standard
*err_msg	pointer to char	Standard
dist1	double	Standard
dist2	double	Standard
flag	unsigned int	Standard
Calls		
Function	Where Described	
print_short_pvparams	Sec. 2.6.7.1.45	

Table 2.6-354: print_error CSU [?????]

2.6.7.1.47 print_debug_list CSU

This CSU writes to a file a list of hit points, exiting with condition code 1 if it finds a cell pointing to itself.

Parameters		
Parameters	Type	Where Typedef Declared
*hitp	pointer to PV_HIT_PT	Sec. 2.6.7.4
Calls		
Function	Where Described	
print_hitpt	Sec. 2.6.7.1.42	

Table 2.6-355: print_debug_list CSU [?????]

2.6.7.1.48 attenuation_factor CSU

This CSU computes the attenuation factor for radio transmissions partially blocked by terrain or obstacles.

Parameters		
Parameters	Type	Where Typedef Declared
distance	double	Standard
frequency	double	Standard
ReturnValues		
Return Value	Type	Meaning
1.0	static double	Wavelength or distance near zero.
4.0 + distance*195	static double	distance < 0.01
5.12 + distance*83.5	static double	0.01 <= distance < 0.09
5.86 + distance*75.23	static double	0.09 <= distance < 1.0
2.47 + distance*78.629	static double	1.0 <= distance

Table 2.6-356: attenuation_factor CSU [8.6.7.1.31]

2.6.7.1.49 print_guard_info CSU

This CSU writes to a file patch guard information from a PATCH_GUARD data structure.

Parameters		
Parameters	Type	Where Typedef Declared
*patch_guardp	pointer to PATCH_GUARD	terrain.h
patch_num	int	Standard

Table 2.6-357: print_guard_info CSU [?????]

2.6.7.1.50 print_patch_info CSU

This CSU writes to a file patch information from EDGE_DESCRIPTOR and TERRAIN_POINT data structures. This CSU has no parameters, returns no values, and calls no other CSUs.

2.6.7.1.51 skyline_error CSU

This CSU writes to a file skyline error information including visibility codes and hit points, and calls print_pvparams to write visibility data.

Parameters		
Parameters	Type	Where Typedef Declared
oldvis	int	Standard
curvis	int	Standard
denom	double	Standard
*ohitp	pointer to PV_HIT_PT	Sec. 2.6.7.4
*nhitp	pointer to PV_HIT_PT	Sec. 2.6.7.4
out_pt	int	Standard
Errors		
Error Name	Reason for Error	
"can't open file pve_sky_errs"	Unable to open file pve_sky_errs for appending.	
Calls		
Function	Where Described	
vis_code	Sec. 2.6.7.1.53	
print_pvparams	Sec. 2.6.7.1.44	

Table 2.6-358: skyline_error CSU [?????]

2.6.7.1.52 print_end_info CSU

This CSU writes to a file wrapup information on pv parameters, ratios, visibilities, and attenuation.

Errors	
Error Name	Reason for Error
"can't open file pve_sky_errs"	Unable to open file pve_sky_errs for appending.
"error: returns ..., seg_vis ..., in:"	Segmented visibility is different from pt-to-pt visibility and different from pt-to-pt visibility with 4 bit on.
Calls	
Function	Where Described
print_pvparams	Sec. 2.6.7.1.44
attenuation_factor	Sec. 2.6.7.1.48

Table 2.6-359: print_end_info CSU [?????]

2.6.7.1.53 vis_code CSU

This CSU converts an integer visibility *code* to a character string.

Parameters		
Parameters	Type	Where Typedef Declared
code	int	Standard
Return Values		
Return Value	Type	Meaning
"FULLY VISIBLE"	pointer to char	code = 15
"PT_BLK SOLID"	pointer to char	code = 13
"PT_BLK TREES"	pointer to char	code = 7
"PT_BLK_SOLID_PT_BLK_TREES"	pointer to char	code = 5
"FULL_BLK TREES"	pointer to char	code = 3
"PT_BLK_SOLID_FULL_BLK_TREES"	pointer to char	code = 1
"INVISIBLE"	pointer to char	code = 0
"Unknown vis code"	pointer to char	Unknown code.

Table 2.6-360: vis_code CSU [?????]

2.6.7.1.54 report_vis_change CSU

This CSU writes to a file the present visibility.

Parameters		
Parameters	Type	Where Typedef Declared
vis	int	Standard

Table 2.6-361: report_vis_change CSU [?????]

2.6.7.1.55 get_object_name CSU

This CSU converts an integer *object_type* to a character string.

Parameters		
Parameters	Type	Where Typedef Declared
object_type	int	Standard

Return Values		
Return Value	Type	Meaning
"church"	pointer to static char	object_type = CHURCH
"guard tower"	pointer to static char	object_type = GUARD_TOWER
"house"	pointer to static char	object_type = HOUSE
"mobile home"	pointer to static char	object_type = MOBILE_HOME
"office building"	pointer to static char	object_type = OFFICE_BUILDING
"one story barracks"	pointer to static char	object_type = ONE_STORY_BARRACKS
"power tower"	pointer to static char	object_type = POWER_TOWER
"small house"	pointer to static char	object_type = SMALL_HOUSE
"two story barracks"	pointer to static char	object_type = TWO_STORY_BARRACKS
"water tower"	pointer to static char	object_type = WATER_TOWER
"object ..."	pointer to static char	object_type is none of the above.

Table 2.6-362: get_object_name CSU [?????]

2.6.7.1.56 interpolate CSU

This CSU returns val_mid, interpolating from val_1 at u_1 and val_2 at u_2. It works only when u_1 and u_2 differ by more than EPSILON.

Parameters		
Parameters	Type	Where Typedef Declared
u_1	register double	Standard
u_2	register double	Standard
u_mid	register double	Standard
val_1	register double	Standard
val_2	register double	Standard
ReturnValues		
Return Value	Type	Meaning
$(val_1 + val_2) / 2$	double	u_1 and u_2 are nearly equal.
$(val_2 * (u_{mid} - u_1) + val_1 * (u_2 - u_{mid})) / (u_2 - u_1)$	double	All other conditions.

Table 2.6-363: interpolate CSU [?????]

2.6.7.1.57 print_terrain_point CSU

This CSU writes to a file x, y, and z from a TERRAIN_POINT data structure to a file.

Parameters		
Parameters	Type	Where Typedef Declared
*fp	pointer to FILE	Standard
*pointp	pointer to TERRAIN_POINT	terrain.h

Table 2.6-364: print_terrain_point CSU [?????]

2.6.7.2 pv_bv.c CSC

/simnet/libsrc/libpvis/pv_bv.c

This CSC contains two CSUs, described below.

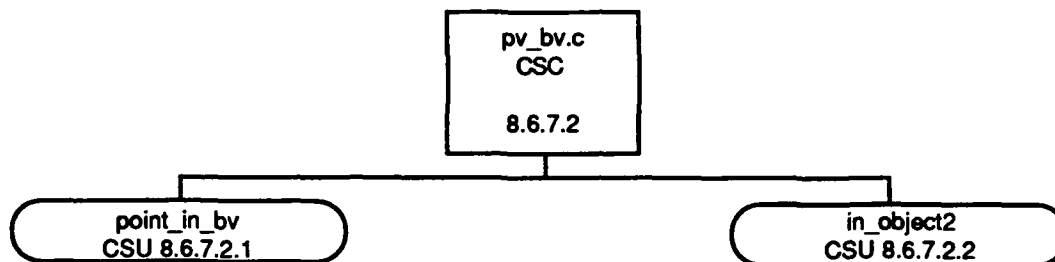


Figure 2.6-26: pv_bv.c CSC Structure

In addition, the CSC contains a single constant definition in the following table.

Constant	Value
OBSTACLE_EXPANSION	3.0 /* meters to add to obstacle */

Table 2.6-365: pv_bv.c Constant Definition

2.6.7.2.1 point_in_bv CSU

Parameters		
Parameters	Type	Where Typedef Declared
*position	pointer to double	Standard
ReturnValues		
Return Value	Type	Meaning
TRUE	int	Objects present.
FALSE	int	No objects to start.
Errors		
Error Name	Reason for Error	
"tdb_get_terrain error at ..."	A call to function tdb_get_terrain returned NULL.	

Calls	
Function	Where Described
tdb get terrain	Sec. 2.21.7.18.1
GET OBJECT MAX X	terrain.h
GET OBJECT MIN X	terrain.h
GET OBJECT MAX Y	terrain.h
GET OBJECT MIN Y	terrain.h

Table 2.6-366: point_in_bv CSU [8.6.7.2.1]

2.6.7.2.2 in_object2 CSU

This CSU is like in_object but includes a check of a grid locator word. This CSU returns 1 if the point (x,y) with grid locator word *glword* is within the North_South_east_West extent of the <num_objects> objects, with OBJECT_DESCRIPTIONS beginning at *objectp*.. It returns 0 otherwise.

Parameters		
Parameters	Type	Where Typedef Declared
num_objects	register int	Standard
*objectp	pointer to register OBJECT_DESCRIPTOR	terrain.h
x	register double	Standard
y	register double	Standard
glword	HWND	mass_std.h
ReturnValues		
Return Value	Type	Meaning
1	int	Point (x,y) with grid locator word within extent of <num_objects> objects.
0	int	Not within.
Calls		
Function	Where Described	
GET OBJECT MAX X	terrain.h	
GET OBJECT MIN X	terrain.h	
GET OBJECT MAX Y	terrain.h	
GET OBJECT MIN Y	terrain.h	

Table 2.6-367: in_object2 CSU [8.6.7.2.2]

2.6.7.3 pve.h CSU

/simnet/libsrc/libpvis/pve.h

This CSU contains constant defines, macro expansions (defined in Appendix A) and three structure definitions.

The constant defines are contained in the following tables.

If TRUE is not defined, the following values are assigned TRUE and FALSE.

Constant	Value
TRUE	1
FALSE	0

Table 2.6-368: TRUE and FALSE Constant Definitions

Constant	Value
EPSI	0.001
TWO EPSI	0.002
EPSI EPSI	0.000001
EPSILON	1.0e-10
THRESHOLD	1.0
PV BIG RATIO	1e+15
PV SMALL RATIO	-1e+15
PV SMALL TREE RATIO	-1e+20
PATCH SIDE	500
TREE MAX Z	15.0
DEFAULT TREELINE WIDTH	1.5
ACCURATE DIFFRAC RANGE	1000

Table 2.6-369: pve.h Size Constant Definitions

Constant	Value
TERMINATE	16
RES OFLOW	6
PV GRID NUM	4
REFINEMENT LEVEL	2
OBJECT VERTS	4
BVOL VERTS	4

Table 2.6-370: pve.h Constant Definitions

Constant	Value
FULL TREE MASK	3 /* set 4 and 8 bits to 0 */
PART TREE MASK	7 /* set 8 bit to 0 */
PART SOLID MASK	13 /* set 2 bit to 0 */
INVIS MASK	0 /* set all bits to 0 */

Table 2.6-371: Visibility Mask Values Constant Definitions

Constant	Value
T LINE BOT WINDOW	1
MIN TREELINE HEIGHT	T LINE BOT WINDOW
TREE TOP CUTOFF	2

Table 2.6-372: Tree and Treeline Height Constant Definitions

Constant	Value
FIR TREE	0xf0f0
SMALL TREE	0xf1f0
LARGE TREE	0xf2f0
TALL BUSH	0xf3f0
SHORT BUSH	0xf4f0
SOLDIER	0xf5f0
ROCK	0xf6f0
GRAVESTONE	0xf7f0
JEEP	0xf2f1
TENT	0xf3f1
MOBILE HOME	0xf4f1
ONE STORY BARRACKS	0xf5f1
HOUSE	0xf6f1
SMALL HOUSE	0xf7f1
OFFICE BUILDING	0xf8f1
GUARD TOWER	0xf9f1
WATER TOWER	0xfbf1
TWO STORY BARRACKS	0xfef1
TELEPHONE	0xffc1
POWER TOWER	0xffd1
CHURCH	0xfff1

Table 2.6-373: Obstacles Constant Definitions

Constant	Value
PVE LEFT	0
PVE TOP	1
PVE RIGHT	2
PVE BOTTOM	3

Table 2.6-374: PVE_Edges Constant Definitions

Constant	Value
PVE GLW LEFT	13
PVE GLW TOP	31
PVE GLW RIGHT	28
PVE GLW BOTTOM	7
PVE CLV ERROR	0

Table 2.6-375: PVE_GLW_Edges Constant Definitions

The following constants define various edge types used in calls to CSU check_edges, Sec. 2.6.7.1.8.

Constant	Value
TERRAIN	0
OBJECT_IN	1
OBJECT_OUT	2
TREE	3
TREELINE	4
TOTAL BLOCK CANOPY	5
PART VIS CANOPY	6

Table 2.6-376: Edge Type Constant Definitions

The first of the three typedef structures is tagged point.

Item	Type	Where Type Defined
x_targ	int	Standard
y_targ	int	Standard
seen	int	Standard
x_hit	int	Standard
y_hit	int	Standard

Table 2.6-377: point tagged Structure Definition

The following typedef struct is tagged pvis_vertex.

Item	Type	Where Type Defined
x	double	Standard
y	double	Standard
z	double	Standard
dist	double	Standard

Table 2.6-378: PV_VERTEX Structure Definition

The following typedef struct is tagged str.

Item	Type	Where Type Defined
minx : 2	HWORD	mass stdc.h
miny : 2	HWORD	mass stdc.h
minz : 2	HWORD	mass stdc.h
maxx : 2	HWORD	mass stdc.h
maxy : 2	HWORD	mass stdc.h
maxz : 2	HWORD	mass stdc.h
waste : 4 /* unused bits */	HWORD	mass stdc.h

Table 2.6-379: PVIS_STR Structure Definition

2.6.7.4 pvis_call.h CSU

/simnet/libsrc/libpvis/pvis_call.h

This CSU contains a number of constant defines including the testing and debugging previously mentioned and contained in the following table.

Constant	Value
COUNT_HITS	0 /* Don't collect stats on number of hits found */
CHECK_DIFFRAC	0 /* Don't check diffraction approximation accuracy */

Table 2.6-380: pvis_call.h Test and Debug Constant Definitions

The remaining constant defines can be broken down into three groups. The first of these groups specifies the functionality of CSU pve_checkvis. The constants can be logically ORed to provide the value of the vis_test field in the PV_PARMS structure.

Constant	Value
PV_SEGMENTS	1
PV_PT_TO_PT	2
PV_DIFFRACTION	4
PV_CLUTTER	8
PV_OBSTACLES	16
PV_AIR_OBSTACLES	(1<<5)
PV_PATCH_HEIGHT	(1<<6)
PV_PATCH_GUARDS_ONLY	(1<<7)
PV_MAX_GROUND_ELEV	(1<<8)

Table 2.6-381: vis_test Constant Definitions For The PV_PARMS Structure

The second group of constants is the visibility code values that CSU pve_checkvis uses. If a visibility check is not made, pve_checkvis returns DUMMY, 8, which is a constant defined in the third, miscellaneous, constant definition table.

Constant	Value
FULLY_VISIBLE	15
PT_BLK_SOLID	13
PT_BLK_TREES	7
PT_BLK_SOLID_PT_BLK_TREES	5
FULL_BLK_TREES	3
PT_BLK_SOLID_FULL_BLK_TREES	1
INVISIBLE	0

Table 2.6-382: pve_checkvis Constant Definition Returns

Constant	Value
NOTALOS	13
CLUTTER	10
DUMMY /* pv checkvis return if no checks */	8
BLOCKED	-1
PV_MAX_HITS_PER_PATCH	100
PV_MAX_TREE_BLOCKS	4

Table 2.6-383: pvis_call.h Miscellaneous Constant Definitions

This header file contains three structures. The first structure, named PV_CHANGE_PT, is contained in the following table.

Item	Type	Where Type Defined
vis	int	Standard
x	double	Standard
y	double	Standard

Table 2.6-384: PV_CHANGE_PT Structure Definition

The element vis gives the visibility on the segment between this PV_CHANGE_PT and the previous one; the other elements are the coordinates of the point.

The second structure, PV_HIT_PT, is tagged hit_point and describes a hit point in an array of forwardly linked hit points.

Item	Type	Where Type Defined
type	int	Standard
x	double	Standard
y	double	Standard
z_top	double	Standard
z_bot_in	double	Standard
z_bot_out	double	Standard
dist	double	Standard
ratio	double	Standard
*nextp	pointer to struct hit_point	This typedef struct

Table 2.6-385: PV_HIT_PT Structure Definition

The "z_" in the above structure are relative to z_from (ie. z_from set as zero), and the z_bot_in and z_bot_out are valid only for trees and objects. For objects, the "ratio" is at the top of the object.

Both of the above structures are contained in the final structure definition, PV_PARAMS, which contains all the parameters for checking visibility in pve_checkvis.

Item	Type	Where Type Defined
vis tests /* ORed vis tests */	int	Standard
eye_pt[4] /* eye pt coords */	double	Standard
targ_pt[4] /* target coords */	double	Standard
guardp / terrain db patch */	pointer to PATCH_GUARD	terrain.h
obj_margin	double	Standard
obstacle_hit	int	Standard
obstacle_min	TDB_POINT	tdb.h
obstacle_max	TDB_POINT	tdb.h
max_change_pts	int	Standard
*change_ptsp	pointer to PV_CHANGE_PT	Sec. 2.6.7.4
max_hit_pts	int	Standard
*hit_ptst	pointer to PV_HIT_PT	Sec. 2.6.7.4
num_change_pts	int	Standard
info	int	Standard
block	TDB_POINT	tdb.h
clutter	TDB_POINT	tdb.h
num_trees	int	Standard
trees[PV_MAX_TREE_BLOCKS]	TDB_POINT	tdb.h
canopy_hit	int	Standard
delta_dist	double	Standard
num_patch_pts	int	Standard
*patch_height_pts	pointer to TDB_POINT	tdb.h
max_elev_pt	TDB_POINT	tdb.h
#if CHECK_DIFFRAC		
exact_diff_range	int	Standard
exact_dist	double	Standard
#endif		
#if COUNT_HITS		
tot_terr_edges	int	Standard
terr_edge_checks	int	Standard
terr_mid_hits	int	Standard
terr_vtx1_hits	int	Standard
terr_vtx2_hits	int	Standard
tot_can_edges	int	Standard
can_edge_checks	int	Standard
can_mid_hits	int	Standard
can_vtx1_hits	int	Standard
can_vtx2_hits	int	Standard
#endif		

Table 2.6-386: PV_PARAMS Structure Definition

2.6.8 libdatabase CSC

/simnet/libsrc/libdatabase

This library handles the loading of and the queries to the parameter files.

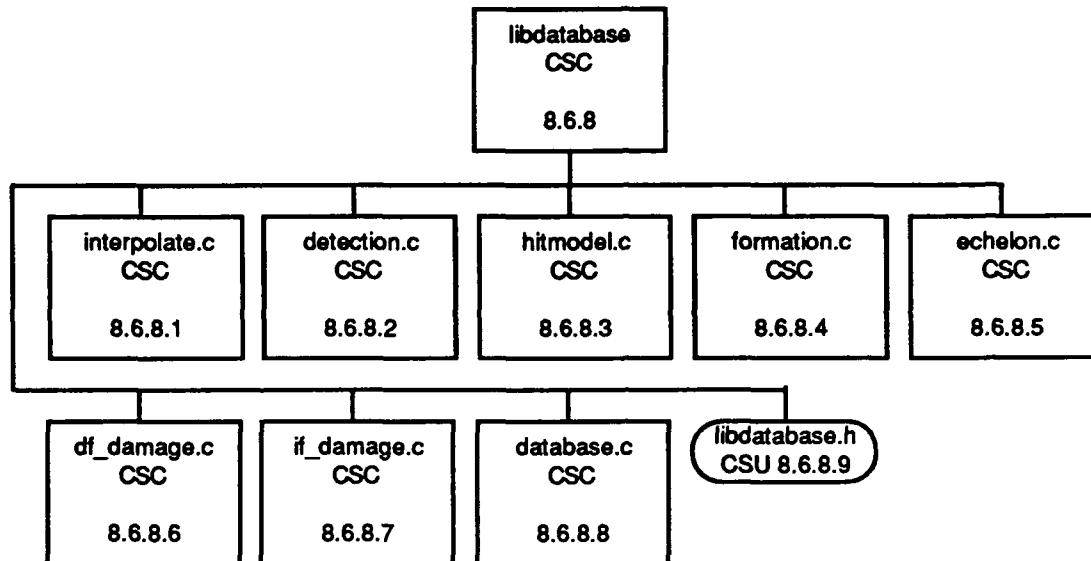


Figure 2.6-26: libdatabase CSC Structure

2.6.8.1 interpolate.c CSC

/simnet/libsrc/libdatabase/interpolate.c

This CSC contains the curve interpolation CSU interpolate_curve.

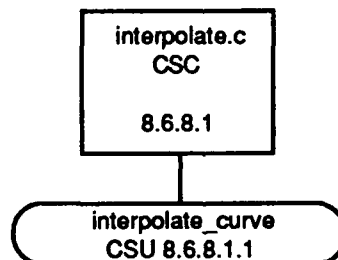


Figure 2.6-27: interpolate.c CSC Structure

2.6.8.1.1 interpolate_curve CSU

Parameters		
Parameters	Type	Where Typedef Declared
*curve	pointer to DATA_UNION	Sec. 2.1.1.5
in_x	REAL	sim_types.h

ReturnValues		
Return Value	Type	Meaning
0.0	REAL	No points.
(curve[1].array)[2].real	REAL	(num_points==1) (in_x<=(curve[1].array)[1].real)
(curve[num_points].array)[2].real	REAL	in_x>=(curve[num_points].array)[1].real
$y1 + ((in_x - x1) * (y2 - y1) / (x2 - x1))$	REAL	(y1 != y2) (x2 != x1)
y1	REAL	None of the above occurs.

Table 2.6-362: interpolate_curve CSU [8.6.8.1.1]

2.6.8.2 detection.c CSC

/simnet/libsrc/libdatabase/detection.c

This CSC contains detection database functions.

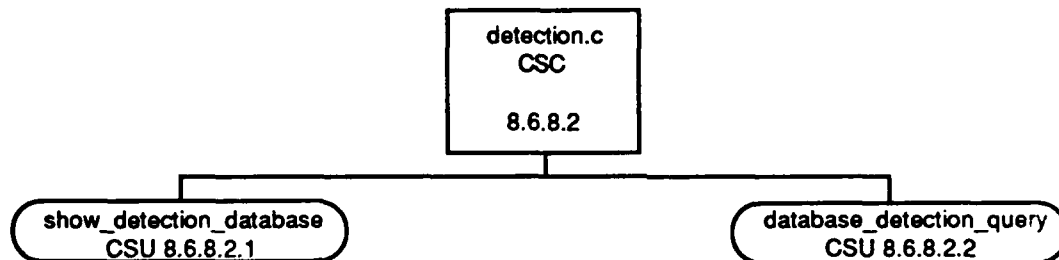


Figure 2.6-28: detection.c CSC Structure

2.6.8.2.1 show_detection_database CSU

This CSU displays primary and secondary curves.

Parameters		
Parameters	Type	Where Typedef Declared
*db	pointer to DATA UNION	Sec. 2.1.1.5
Calls		
Function	Where Described	
show_curves	Sec. 2.6.8.8.3	

Table 2.6-363: show_detection_database CSU [8.6.8.2.1]

2.6.8.2.2 database_detection_query CSU

Parameters		
Parameters	Type	Where Typedef Declared
del_db	DATA UNION	Sec. 2.1.1.5
view_type	int	Standard
pri_sec	int	Standard
range	REAL	sim_types.h
view	int	Standard
ReturnValues		
Return Value	Type	Meaning
interpolate_curve(...)	REAL	Normal end.
Calls		
Function	Where Described	
interpolate_curve	Sec. 2.6.8.1.1	

Table 2.6-364: database_detection_query CSU [8.6.8.2.2]

2.6.8.3 hitmodel.c CSC

/simnet/libsrc/libdatabase/hitmodel.c

This CSC contains the hitmodel database CSU.

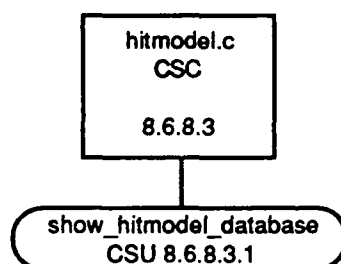


Figure 2.6-29: hitmodel.c CSC Structure

2.6.8.3.1 show_hitmodel_database CSU

This CSU displays hitmodel curves.

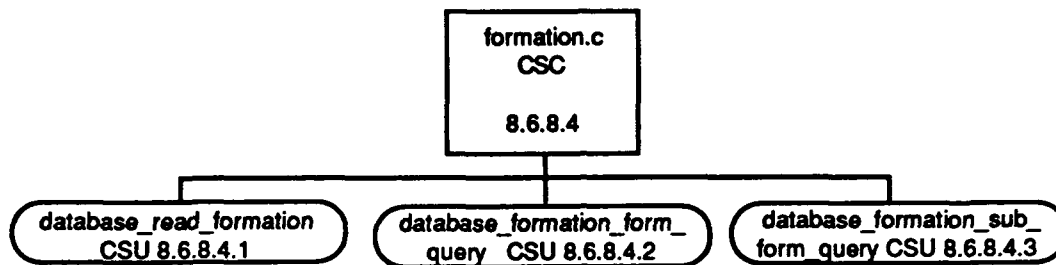
Parameters		
Parameters	Type	Where Typedef Declared
*db	pointer to DATA UNION	Sec. 2.1.1.5
Calls		
Function	Where Described	
show_curves	Sec. 2.6.8.8.3	

Table 2.6-365: show_hitmodel_database CSU [8.6.8.3.1]

2.6.8.4 formation.c CSC

/simnet/libsrc/libdatabase/formation.c

This CSC contains formation database CSUs.

**Figure 2.6-30: formation.c CSC Structure****2.6.8.4.1 database_read_formation CSU**

This CSU attempts to read a database file and displays a message if unable to.

Parameters		
Parameters	Type	Where Typedef Declared
tactics	int	Standard
*name	pointer to char	Standard
Calls		
Function	Where Described	
reader_read_file	Sec. 2.1.1.1.7	

Table 2.6-366: database_read_formation CSU [8.6.8.4.1]**2.6.8.4.2 database_formation_form_query CSU**

Parameters		
Parameters	Type	Where Typedef Declared
tactics	int	Standard
*echelon	pointer to char	Standard
*formation	pointer to char	Standard
*job	pointer to char	Standard
ReturnValues		
Return Value	Type	Meaning
find_tag(...)	pointer to DATA_UNION	Normal end.
Calls		
Function	Where Described	
find_tag	Sec. 2.1.1.4.3	
get_symbol	Sec. 2.1.1.3.2	

Table 2.6-367: database_formation_form_query CSU [8.6.8.4.2]

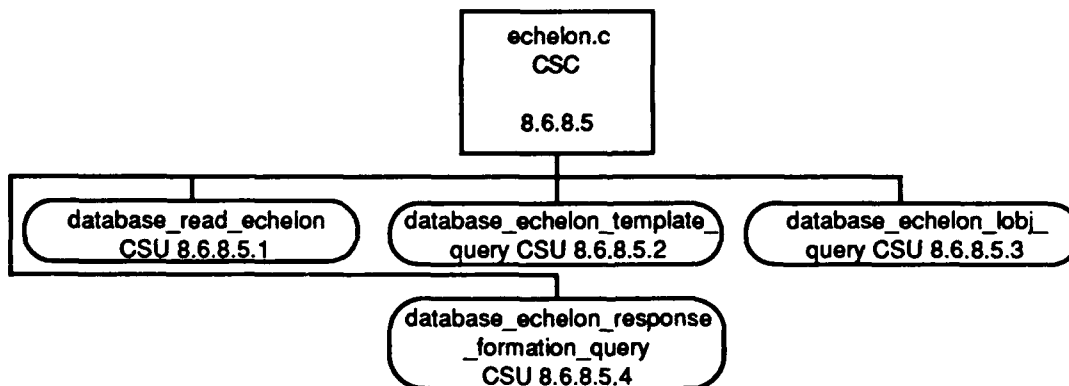
2.6.8.4.3 database_formation_sub_form_query CSU

Parameters		
Parameters	Type	Where Typedef Declared
tactics	int	Standard
*echelon	pointer to char	Standard
*formation	pointer to char	Standard
*job	pointer to char	Standard
ReturnValues		
Return Value	Type	Meaning
db[2].array	pointer to DATA UNION	Normal end.
Calls		
Function	Where Described	
find tag	Sec. 2.1.1.4.3	
get symbol	Sec. 2.1.1.3.2	
database_formation_form_query	Sec. 2.6.8.4.2	

Table 2.6-368: database_formation_sub_form_query CSU [8.6.8.4.3]**2.6.8.5 echelon.c CSC**

/simnet/libsrc/libdatabase/echelon.c

This CSC contains echelon database CSUs.

**Figure 2.6-31: echelon.c CSC Structure**

2.6.8.5.1 database_read_echelon CSU

This CSU attempts to read a database file and displays a message if unable.

Parameters		
Parameters	Type	Where Typedef Declared
tactics	int	Standard
*name	pointer to char	Standard
Calls		
Function	Where Described	
reader_read_file	Sec. 2.1.1.1.7	

Table 2.6-369: database_read_echelon CSU [8.6.8.5.1]

2.6.8.5.2 database_echelon_template_query CSU

Parameters		
Parameters	Type	Where Typedef Declared
tactics	int	Standard
va_alist	varargs	Standard
ReturnValues		
Return Value	Type	Meaning
0	pointer to DATA_UNION	Keyword or template not found.
hold	pointer to DATA_UNION	Normal end.
Calls		
Function	Where Described	
get_symbol	Sec. 2.1.1.3.2	
find_tag	Sec. 2.1.1.4.3	

Table 2.6-370: database_echelon_template_query CSU [8.6.8.5.2]

2.6.8.5.3 database_echelon_lobj_query CSU

Parameters		
Parameters	Type	Where Typedef Declared
tactics	int	Standard
*echelon	pointer to char	Standard
*parm_name	pointer to char	Standard
ReturnValues		
Return Value	Type	Meaning
database_echelon_template_query(...)	pointer to DATA_UNION	Normal end.

Calls	
Function	Where Described
database_echelon_template_query	Sec. 2.6.8.5.2

Table 2.6-371: database_echelon_lobj_query CSU [8.6.8.5.3]

2.6.8.5.4 database_echelon_response_formation_query CSU

Parameters		
Parameters	Type	Where Typedef Declared
tactics	int	Standard
*echelon	pointer to char	Standard
*context	pointer to char	Standard
ReturnValues		
Return Value	Type	Meaning
result	pointer to DATA_UNION	Formation is context sensitive.
hold	pointer to DATA_UNION	Formation is not context sensitive.
Calls		
Function	Where Described	
database_echelon_template_query	Sec. 2.6.8.5.2	
find_tag	Sec. 2.1.1.4.3	
get_symbol	Sec. 2.1.1.3.2	

Table 2.6-372: database_echelon_response_formation_query CSU [8.6.8.5.4]

2.6.8.6 df_damage.c CSC

/simnet/libsrc/libdatabase/df_damage.c

This CSC contains the direct fire database CSUs and constants that define vehicle walls.

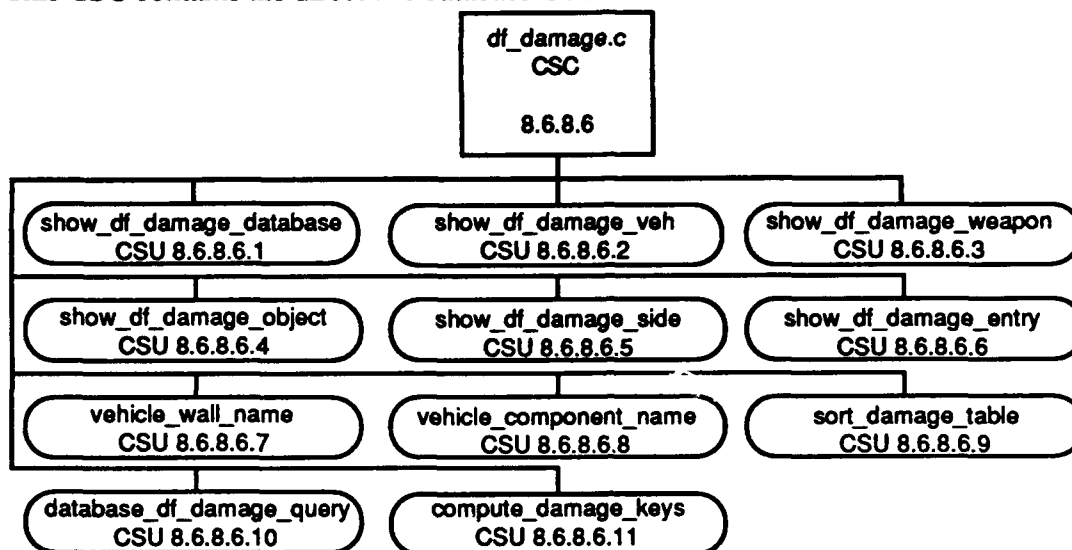


Figure 2.6-32: df_damage.c CSC Structure

The constant definitions for this CSC are in the following table.

Constant	Value
VEHICLE WALL FRONT	1
VEHICLE WALL BACK	2
VEHICLE WALL LEFT	3
VEHICLE WALL RIGHT	4
VEHICLE WALL TOP	5

Table 2.6-373: df_damage.c Constant Definitions

2.6.8.6.1 show_df_damage_database CSU

This CSU displays information from the direct fire damage database.

Parameters		
Parameters	Type	Where Typedef Declared
df_damage	DATA UNION	Sec. 2.1.1.5
Calls		
Function	Where Described	
show_df_damage_veh	Sec. 2.6.8.6.2	
find_tag	Sec. 2.1.1.4.3	
get_symbol	Sec. 2.1.1.3.2	

Table 2.6-374: show_df_damage_database CSU [8.6.8.6.1]

2.6.8.6.2 show_df_damage_veh CSU

Parameters		
Parameters	Type	Where Typedef Declared
*db	pointer to DATA_UNION	Sec. 2.1.1.5
Calls		
Function	Where Described	
show df damage weapon	Sec. 2.6.8.6.3	
find tag	Sec. 2.1.1.4.3	
get symbol	Sec. 2.1.1.3.2	

Table 2.6-375: show_df_damage_veh CSU [8.6.8.6.2]**2.6.8.6.3 show_df_damage_weapon CSU**

Parameters		
Parameters	Type	Where Typedef Declared
*db	pointer to DATA_UNION	Sec. 2.1.1.5
Calls		
Function	Where Described	
show df damage object	Sec. 2.6.8.6.4	

Table 2.6-376: show_df_damage_weapon CSU [8.6.8.6.3]**2.6.8.6.4 show_df_damage_object CSU**

Parameters		
Parameters	Type	Where Typedef Declared
*db	pointer to DATA_UNION	Sec. 2.1.1.5
Calls		
Function	Where Described	
show df damage side	Sec. 2.6.8.6.5	
find tag	Sec. 2.1.1.4.3	
get symbol	Sec. 2.1.1.3.2	

Table 2.6-377: show_df_damage_object CSU [8.6.8.6.4]**2.6.8.6.5 show_df_damage_side CSU**

Parameters		
Parameters	Type	Where Typedef Declared
*db	pointer to DATA_UNION	Sec. 2.1.1.5

Calls	
Function	Where Described
show_df_damage_entry	Sec. 2.6.8.6.6

Table 2.6-378: show_df_damage_side CSU [8.6.8.6.5]

2.6.8.6.6 show_df_damage_entry CSU

Parameters		
Parameters	Type	Where Typedef Declared
*db	pointer to DATA UNION	Sec. 2.1.1.5

Table 2.6-379: show_df_damage_entry CSU [8.6.8.6.6]

2.6.8.6.7 vehicle_wall_name CSU

This CSU converts the value for a vehicle wall name to a character string.

Parameters		
Parameters	Type	Where Typedef Declared
side	int	Standard
ReturnValues		
Return Value	Type	Meaning
"front"	pointer to char	VEHICLE WALL FRONT
"back"	pointer to char	VEHICLE WALL BACK
"left"	pointer to char	VEHICLE WALL LEFT
"right"	pointer to char	VEHICLE WALL RIGHT
"top"	pointer to char	VEHICLE WALL TOP
"unknown"	pointer to char	Name unknown.

Table 2.6-380: vehicle_wall_name CSU [8.6.8.6.7]

2.6.8.6.8 vehicle_component_name CSU

This CSU converts the value for a vehicle component name to a character string.

Parameters		
Parameters	Type	Where Typedef Declared
component	VehicleComponent	Standard
ReturnValues		
Return Value	Type	Meaning
"hull"	pointer to char	hullComponent
"turret"	pointer to char	turretComponent
"unknown"	pointer to char	Component unknown.

Table 2.6-381: vehicle_component_name CSU [8.6.8.6.8]

2.6.8.6.9 sort_damage_table CSU

Parameters		
Parameters	Type	Where Typedef Declared
*db	pointer to DATA UNION	Sec. 2.1.1.5
Calls		
Function	Where Described	
sort_tag_table	Sec. 2.1.1.4.5	

Table 2.6-382: sort_damage_table CSU [8.6.8.6.9]

2.6.8.6.10 database_df_damage_query CSU

Parameters		
Parameters	Type	Where Typedef Declared
*db	pointer to DATA UNION	Sec. 2.1.1.5
*db_map	pointer to DATA UNION	Sec. 2.1.1.5
random_number	int	Standard
ammo	ObjectType	p_sim.h
fireresult	FireResult	p_sim.h
component	VehicleComponent	basic.h
*impact	pointer to float	Standard
*trajectory	pointer to float	Standard
*dimensions	REAL	sim_types.h
debug_flag	int	Standard
ReturnValues		
Return Value	Type	Meaning
NO_DAMAGE	int	Unknown munition, unknown component, unknown angle, proximity fuze outside range, no catastrophic damage, no mobility damage, or no firepower damage.
Calls		
Function	Where Described	
find_tag_sorted	Sec. 2.1.1.4.7	
compute_damage_keys	Sec. 2.6.8.6.11	
vehicle_component_name	Sec. 2.6.8.6.8	
vehicle_wall_name	Sec. 2.6.8.6.7	
show_df_damage_entry	Sec. 2.6.8.6.6	
vec_mag3	sim_macros.h	

Table 2.6-383: database_df_damage_query CSU [8.6.8.6.10]

2.6.8.6.11 compute_damage_keys CSU

Prior to this CSU, two constants are defined (#define TAN30 0.57735; #define TAN60 1.73205) and a static side_lookup_table is declared and initialized.

Parameters		
Parameters	Type	Where Typedef Declared
*impact	pointer to float	Standard
*trajectory	pointer to float	Standard
*dimensions	pointer to REAL	sim_types.h
*side	pointer to int	Standard
*angle	pointer to int	Standard
Calls		
Function	Where Described	
fvec to rvec	Sec. 2.14.3.5.13	
which_side	Sec. 2.14.3.5.9	

Table 2.6-384: compute_damage_keys CSU [8.6.8.6.11]

2.6.8.7 if_damage.c CSC

/simnet/libsrc/libdatabase/if_damage.c

This CSC contains the indirect fire database CSUs.

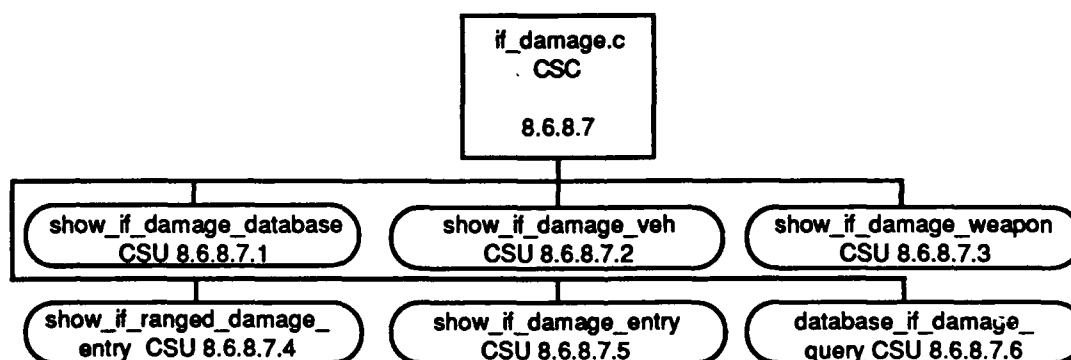


Figure 2.6-33: if_damage.c CSC Structure

2.6.8.7.1 show_if_damage_database CSU

Parameters		
Parameters	Type	Where Typedef Declared
if damage	DATA UNION	Sec. 2.1.1.5
Calls		
Function	Where Described	
show_if_damage_veh	Sec. 2.6.8.7.2	

Table 2.6-385: show_if_damage_database CSU [8.6.8.7.1]

2.6.8.7.2 show_if_damage_veh CSU

Parameters		
Parameters	Type	Where Typedef Declared
*db	pointer to DATA_UNION	Sec. 2.1.1.5
Calls		
Function	Where Described	
show if damage weapon	Sec. 2.6.8.7.3	

Table 2.6-386: show_if_damage_veh CSU [8.6.8.7.2]**2.6.8.7.3 show_if_damage_weapon CSU**

Parameters		
Parameters	Type	Where Typedef Declared
*db	pointer to DATA_UNION	Sec. 2.1.1.5
Calls		
Function	Where Described	
show_if_ranged_damage_entry	Sec. 2.6.8.7.4	

Table 2.6-387: show_if_damage_weapon CSU [8.6.8.7.3]**2.6.8.7.4 show_if_ranged_damage_entry CSU**

Parameters		
Parameters	Type	Where Typedef Declared
*db	pointer to DATA_UNION	Sec. 2.1.1.5
Calls		
Function	Where Described	
show if damage entry	Sec. 2.6.8.7.5	

Table 2.6-388: show_if_ranged_damage_entry CSU [8.6.8.7.4]**2.6.8.7.5 show_if_damage_entry CSU**

Parameters		
Parameters	Type	Where Typedef Declared
*db	pointer to DATA_UNION	Sec. 2.1.1.5

Table 2.6-389: show_if_damage_entry CSU [8.6.8.7.5]

2.6.8.7.6 database_if_damage_query CSU

Parameters		
Parameters	Type	Where Typedef Declared
*db	pointer to DATA UNION	Sec. 2.1.1.5
*db_map	pointer to DATA UNION	Sec. 2.1.1.5
rand	int	Standard
ammo	ObjectType	p sim.h
fuze	ObjectType	p sim.h
r2	REAL	sim_types.h
debug_flag	int	Standard
ReturnValues		
Return Value	Type	Meaning
NO_DAMAGE	int	Unknown fuze, unknown ammunition, w table overrun, or none of the damage below.
CATASTROPHIC_DAMAGE	int	Catastrophic damage selected randomly.
MOBILITY_DAMAGE	int	Mobility damage selected randomly.
FIREPOWER_DAMAGE	int	Firepower damage selected randomly.
Calls		
Function	Where Described	
find_tag_sorted	Sec. 2.1.1.4.7	
find_tag	Sec. 2.1.1.4.3	
show_if_damage_entry	Sec. 2.6.8.7.5	

Table 2.6-390: database_if_damage_query CSU [8.6.8.7.6]

2.6.8.8 database.c CSC

/simnet/libsrc/libdatabase/database.c

This CSC contains the common database utility CSUs.

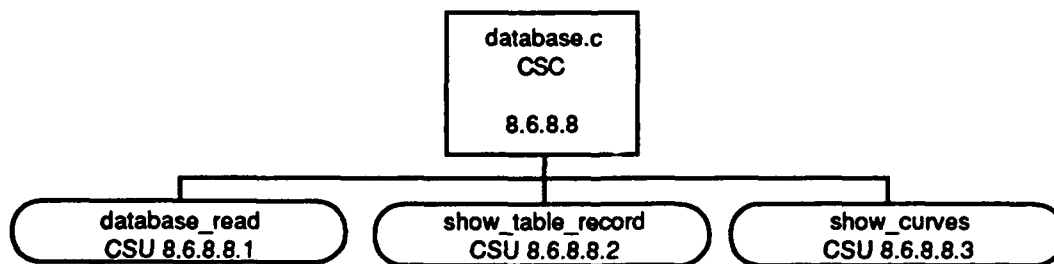


Figure 2.6-34: database.c CSC Structure

2.6.8.8.1 database_read CSU

This CSU attempts to read a database file and displays a message if unable.

Parameters		
Parameters	Type	Where Typedef Declared
*db	pointer to DATA UNION	Sec. 2.1.1.5
*name	pointer to char	Standard
Calls		
Function	Where Described	
reader_read_file	Sec. 2.1.1.7	

Table 2.6-391: database_read CSU [8.6.8.8.1]

2.6.8.8.2 show_table_record CSU

Parameters		
Parameters	Type	Where Typedef Declared
*table	pointer to DATA UNION	Sec. 2.1.1.5

Table 2.6-392: show_table_record CSU [8.6.8.8.2]

2.6.8.8.3 show_curves CSU

Parameters		
Parameters	Type	Where Typedef Declared
*curves	pointer to DATA UNION	Sec. 2.1.1.5
Calls		
Function	Where Described	
show_table_record	Sec. 2.6.8.8.2	

Table 2.6-393: show_curves CSU [8.6.8.8.3]

2.6.8.9 libdatabase.h CSU

/simnet/libsrc/libdatabase/libdatabase.h

This CSU contains constants, macros, and external function definitions. The macro `data_base_hitmodel_query` is shown in Appendix A and the constants are shown in the following table.

Constant	Value
VIEW BLOCKED	-1
STATIONARY	0
MOVING	1
HULL DOWN	2
PRIMARY ARC	0 /* Detection parameters */
SECONDARY ARC	1
GROUND VIEW	0
AIR VIEW	1
TACTICS NATO	1 /* Tactics */
TACTICS WARSAW	2
NO DAMAGE	0 /* Damages */
MOBILITY DAMAGE	1
FIREPOWER DAMAGE	2
CATASTROPHIC DAMAGE	3
ARTY TYPE GROUND	1 /* Artillery types */
ARTY TYPE VEHICLE	2
ARTY TYPE DEATH	3

Table 2.6-394: libdatabase.h Constant Definitions

2.6.9 Weapons CSC

The weapons code is called by the local vehicles while executing their ticks. This code simulates the target selection, weapons selection, target tracking, and firing of the missiles. It also determines if the weapon hit its target.

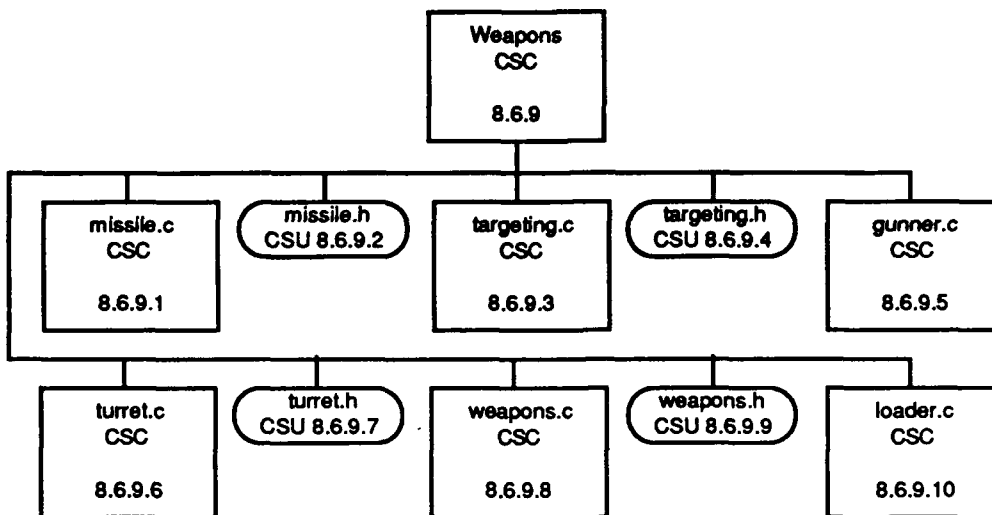


Figure 2.6-35: Weapons CSC Structure

2.6.9.1 missile.c CSC

/simnet/src/host/missile.c

This CSC handles missiles, including creation, deletion, flying the missile, impact, and hit/miss computation.

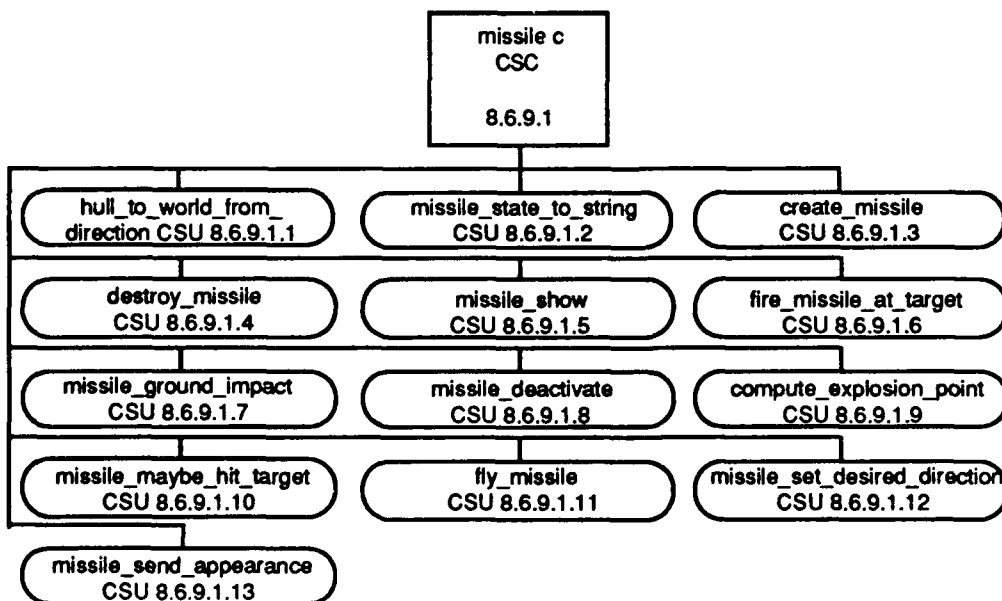


Figure 2.6-36: missile.c CSC Structure

2.6.9.1.1 hull_to_world_from_direction CSU

Parameters		
Parameters	Type	Where Typedef Declared
direction	VECTOR	sim_types.h
h2w	T MATRIX	sim_types.h

Table 2.6-395: hull_to_world_from_direction CSU [8.6.9.1.1]**2.6.9.1.2 missile_state_to_string CSU**

This CSU converts a missile state variable to a character string.

Parameters		
Parameters	Type	Where Typedef Declared
state	int	Standard
ReturnValues		
Return Value	Type	Meaning
"idle"	pointer to char	case MISSILE STATE IDLE
"tracking"	pointer to char	case MISSILE STATE TRACKING
"armed"	pointer to char	case MISSILE STATE ARMED
"lost target"	pointer to char	case MISSILE_STATE_TARGET_LOST
"crashing"	pointer to char	case MISSILE STATE CRASHING
"unknown"	pointer to char	Unknown case.

Table 2.6-396: missile_state_to_string CSU [8.6.9.1.2]**2.6.9.1.3 create_missile CSU**

Parameters		
Parameters	Type	Where Typedef Declared
*table	pointer to DATA UNION	Sec. 2.1.1.5
ownerID	unsigned short	Standard
forceID	ForceID	basic.h
ReturnValues		
Return Value	Type	Meaning
missile	pointer to MISSILE_VARS	Created missile.

Calls	
Function	Where Described
allocate_missile	Sec. 2.6.9.2 See Appendix A
generate_vehicle_id	
get_guises	Sec. 2.6.1.1.8
vec_init	Sec. 2.6.2.61.1 Vehicles CSCI SDD
ft_float	Sec. 2.14.1.2.12
deg_to_rad	sim_macros.h
PrepareDiscrepancyThresholds	
buffer_allocate	Sec. 2.14.4.2.12

Table 2.6-397: create_missile CSU [8.6.9.1.3]

2.6.9.1.4 destroy_missile CSU

This CSU deallocates previously allocated missile memory space.

Parameters		
Parameters	Type	Where Typedef Declared
*missile	pointer to MISSILE_VARS	Sec. 2.6.9.2
Calls		
Function	Where Described	
buffer_deallocate	Sec. 2.14.4.2.15	
deallocate_missile	Sec. 2.6.9.2 See Appendix A	

Table 2.6-398: destroy_missile CSU [8.6.9.1.4]

2.6.9.1.5 missile_show CSU

This CSU displays descriptive information for a missile in flight.

Parameters		
Parameters	Type	Where Typedef Declared
*missile	pointer to MISSILE_VARS	Sec. 2.6.9.2
Calls		
Function	Where Described	
simnet_id_string_from_saf_id		
mps_to_kph	Sec. 2.13.3.1 See Appendix A	
rad_to_deg	sim_macros.h	
missile_state_to_string	Sec. 2.6.9.1.2	

Table 2.6-399: missile_show CSU [8.6.9.1.5]

2.6.9.1.6 fire_missile_at_target CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*weapon	pointer to WEAPON_VARS	Sec. 2.9.1.2
targetID	unsigned short	Standard
*target_pos	pointer to REAL	sim_types.h
*target_vel	pointer to REAL	sim_types.h
Errors		
Error Name	Reason for Error	
"Internal error: Shot a missile that wasn't available"	missile->state != MISSILE_STATE_IDLE	
Calls		
Function	Where Described	
OBJ_POSITION	Sec. 2.9.1.1 See Appendix A	
muzzle_position_in_world_coordinates	Sec. 2.6.9.8.18	
vec_copy	Sec. 2.6.2.59.1 Vehicles CSCI SDD	
vec_sub	Sec. 2.6.2.65.1 Vehicles CSCI SDD	
vec_mag3	sim_macros.h	
vec_scale	Sec. 2.6.2.64.1 Vehicles CSCI SDD	
vec_add	Sec. 2.6.2.57.1 Vehicles CSCI SDD	
OBJ_VELOCITY	Sec. 2.9.1.1 See Appendix A	
vec_normalize	Sec. 2.6.2.63.1 Vehicles CSCI SDD	
hull_to_world_from_direction	Sec. 2.6.9.1.1	
saf_vehicle_next_event_id	Sec. 2.6.1.1.28	
simnet_send_fire		
missile_send_appearance	Sec. 2.6.9.1.13	
DEBUG_MISSILE	Sec. 2.5.2.2 See Appendix A	

Table 2.6-400: fire_missile_at_target CSU [8.6.9.1.6]

2.6.9.1.7 missile_ground_impact CSU

Parameters		
Parameters	Type	Where Typedef Declared
*weapon	pointer to WEAPON_VARS	Sec. 2.9.1.2
*missile	pointer to MISSILE_VARS	Sec. 2.6.9.2

Calls	
Function	Where Described
LOOKUP POSITION	Sec. 2.9.1.1 See Appendix A
vec sub	Sec. 2.6.2.65.1 Vehicles CSCI SDD
vec dot prod	Sec. 2.6.2.54.1 Vehicles CSCI SDD
coords within database	Sec. 2.13.3.1 See Appendix A
simnet send impact	
MOMENTUM	Sec. 2.14.3.9 See Appendix A
ENERGY	Sec. 2.14.3.9 See Appendix A
DEBUG MISSILE	Sec. 2.5.2.2 See Appendix A
missile deactivate	Sec. 2.6.9.1.8

Table 2.6-401: missile_ground_impact CSU [8.6.9.1.7]

2.6.9.1.8 missile_deactivate CSU

Parameters		
Parameters	Type	Where Typedef Declared
*missile	pointer to MISSILE_VARS	Sec. 2.6.9.2
Calls		
Function	Where Described	
simnet send deactivate		
vec init	Sec. 2.6.2.61.1 Vehicles CSCI SDD	

Table 2.6-402: missile_deactivate CSU [8.6.9.1.8]

2.6.9.1.9 compute_explosion_point CSU

Parameters		
Parameters	Type	Where Typedef Declared
*point	pointer to REAL	sim_types.h
*p1	pointer to REAL	sim_types.h
*p2	pointer to REAL	sim_types.h
*result	pointer to REAL	sim_types.h
Calls		
Function	Where Described	
vec sub	Sec. 2.6.2.65.1 Vehicles CSCI SDD	
vec dot prod	Sec. 2.6.2.54.1 Vehicles CSCI SDD	
vec scale	Sec. 2.6.2.64.1 Vehicles CSCI SDD	
vec add	Sec. 2.6.2.57.1 Vehicles CSCI SDD	

Table 2.6-403: compute_explosion_point CSU [8.6.9.1.9]

2.6.9.1.10 missile_maybe_hit_target CSU

This CSU calculates the explosion point of a missile and determines if it hit the target. It must hit within 5 meters of a vehicle to call it a vehicleImpact. Otherwise it is just a proximateImpact. The impact threshold is defined before this CSU (#define IMPACT_THRESHOLD 5.0).

Parameters		
Parameters	Type	Where Typedef Declared
*weapon	pointer to WEAPON_VARS	Sec. 2.9.1.2
*missile	pointer to MISSILE_VARS	Sec. 2.6.9.2
targetID	unsigned short	Standard
*target_position	REAL	sim_types.h
*to target	REAL	sim_types.h
*relative_velocity	REAL	sim_types.h
ReturnValues		
Return Value	Type	Meaning
FALSE	int	Still closing on target.
TRUE	int	Normal end.
Calls		
Function	Where Described	
vec dot prod	Sec. 2.6.2.54.1 Vehicles CSCI SDD	
compute explosion point	Sec. 2.6.9.1.9	
range squared	Sec. 2.14.3.5.10	
DEBUG MISSILE	Sec. 2.5.2.2 See Appendix A	
LOOKUP POSITION	Sec. 2.9.1.1 See Appendix A	
vec sub	Sec. 2.6.2.65.1 Vehicles CSCI SDD	
vec normalize	Sec. 2.6.2.63.1 Vehicles CSCI SDD	
get i and t from normal	Sec. 2.6.4.5.2	
LOOKUP_HULL_TO_WORLD	Sec. 2.9.1.1 See Appendix A	
LOOKUP_TURRETAZIMUTH	Sec. 2.9.1.1 See Appendix A	
LOOKUP_VEHICLECLASS	Sec. 2.9.1.1 See Appendix A	
simnet send impact		
MOMENTUM	Sec. 2.14.3.9 See Appendix A	
ENERGY	Sec. 2.14.3.9 See Appendix A	
missile deactivate	Sec. 2.6.9.1.8	

Table 2.6-404: missile_maybe_hit_target CSU [8.6.9.1.10]

2.6.9.1.11 fly_missile CSU

This CSU performs the calculations for a missile flight. It updates the position and speed, checks arming status, checks to see if it hit the target, determines if a new direction should be taken, checks to see if it hit the ground, and keeps track of fuel consumption.

Parameters		
Parameters	Type	Where Typedef Declared
*weapon	pointer to WEAPON_VARS	Sec. 2.9.1.2
targetID	unsigned short	Standard

ReturnValues		
Return Value	Type	Meaning
ROUND_DONE_FLYING	int	Missile is in idle state or has hit the target or has hit the ground or is in an unknown state.
ROUND IN FLIGHT	int	Missile is in flight.
Calls		
Function	Where Described	
vec_copy	Sec. 2.6.2.59.1 Vehicles CSCI SDD	
vec_scale	Sec. 2.6.2.64.1 Vehicles CSCI SDD	
vec_add	Sec. 2.6.2.57.1 Vehicles CSCI SDD	
min	Sec. 2.6.7.3 & Sec. 2.13.3.5 See Appendix A	
DEBUG MISSILE	Sec. 2.5.2.2 See Appendix A	
saf_vehicle_est_position	Sec. 2.6.1.1.58	
LOOKUP SAFOBJ	Sec. 2.9.1.1 See Appendix A	
LOOKUP VELOCITY	Sec. 2.9.1.1 See Appendix A	
vec_sub	Sec. 2.6.2.65.1 Vehicles CSCI SDD	
vec_normalize	Sec. 2.6.2.63.1 Vehicles CSCI SDD	
range_squared	Sec. 2.14.3.5.10	
missile_maybe_hit_target	Sec. 2.6.9.1.10	
intervis_can_see_pt_to_pt	Sec. 2.6.5.2.1	
missile_set_desired_direction	Sec. 2.6.9.1.12	
vec_mag3	sim-macros.h	
tdb_get_z	Sec. 2.21.7.16.2	
missile_ground_impact	Sec. 2.6.9.1.7	
hull_to_world_from_direction	Sec. 2.6.9.1.1	
missile_send_appearance	Sec. 2.6.9.1.13	

Table 2.6-405: fly_missile CSU [8.6.9.1.11]

2.6.9.1.12 missile_set_desired_direction CSU

Parameters		
Parameters	Type	Where Typedef Declared
*missile	pointer to MISSILE_VARS	Sec. 2.6.9.2
*target_velocity	pointer to REAL	Standard
*direction_to_target	pointer to REAL	Standard
Calls		
Function	Where Described	
square	sim_macros.h	
vec_scale	Sec. 2.6.2.64.1 Vehicles CSCI SDD	
vec_dot_prod	Sec. 2.6.2.54.1 Vehicles CSCI SDD	
vec_sub	Sec. 2.6.2.65.1 Vehicles CSCI SDD	
vec_normalize	Sec. 2.6.2.63.1 Vehicles CSCI SDD	
vec_add	Sec. 2.6.2.57.1 Vehicles CSCI SDD	

Table 2.6-406: missile_set_desired_direction CSU [8.6.9.1.12]

2.6.9.1.13 missile_send_appearance CSU

Parameters		
Parameters	Type	Where Typedef Declared
*missile	pointer to MISSILE VARS	Sec. 2.6.9.2
Calls		
Function	Where Described	
simnet send appearance		

Table 2.6-407: missile_send_appearance CSU [8.6.9.1.13]**2.6.9.2 missile.h CSU**

/simnet/src/host/missile.h

This CSU contains the symbolic constants, structure definition, and macro definitions (shown in Appendix A) used by the missile code.

Constant	Value
MISSILE STATE IDLE	0
MISSILE STATE TRACKING	1
MISSILE STATE ARMED	2
MISSILE STATE TARGET LOST	3
MISSILE STATE CRASHING	4

Table 2.6-408: missile.h Constant Definitions

The following typedef struct is tagged missile_vars.

Item	Type	Where Type Defined
state /* Flying state of missile */	int	Standard
vehicleID /* Appearance of */	unsigned short	Standard
ownerID /* flying missile */	unsigned short	Standard
vehicleClass	VehicleClass	basic.h
appearance	int	Standard
guises	VehicleGuises	basic.h
marking	VehicleMarking	basic.h
capabilities	VehicleCapabilities	basic.h
forceID	ForceID	basic.h
model base adjustment	REAL	sim_types.h
last tick time	unsigned int	Standard
direction /* Physical */	VECTOR	sim_types.h
position /* dynamics */	VECTOR	sim_types.h
velocity	VECTOR	sim_types.h
old position	VECTOR	sim_types.h
hull to world	T MATRIX	sim_types.h
range remaining	REAL	sim_types.h
speed mps	REAL	sim_types.h
desired direction /* Maneu- */	VECTOR	sim_types.h
desired velocity /* ver */	VECTOR	sim_types.h
max vehicle range /* Flying */	REAL	sim_types.h
max speed mps /* & */	REAL	sim_types.h
acceleration /* triggering */	REAL	sim_types.h
max turn rps /* parameters */	REAL	sim_types.h
tan superelevation angle	REAL	sim_types.h
cos max angle to target	REAL	sim_types.h
fuze distance2	REAL	sim_types.h
effective distance2	REAL	sim_types.h
dimensions /* Appearance */	VECTOR	sim_types.h
update thresholds /* packet */	DiscrepancyThresholds	libapp.h
last update / & threshold */	pointer to SimulationPDU	p_sim.h
time at last update /* stuff */	int	Standard

Table 2.6-409: MISSILE_VARS Structure Definition

2.6.9.3 targeting.c CSC

/simnet/src/host/targeting.c

This file contains the code which handles the tracking, prioritizing, and selection of targets.

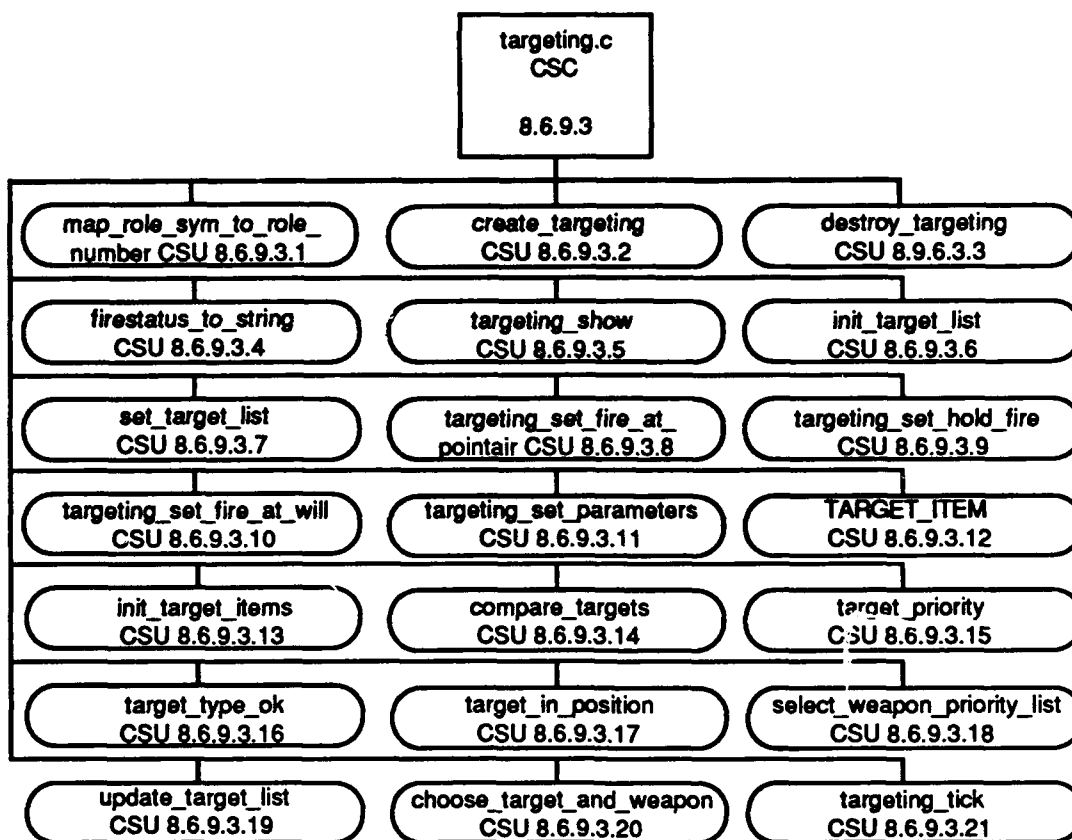


Figure 2.6-37: targeting.c CSC Structure

2.6.9.3.1 map_role_sym_to_role_number CSU

This CSU converts a character role symbol to an integer role number.

Parameters		
Parameters	Type	Where Typedef Declared
*sym	pointer to char	Standard
ReturnValues		
Return Value	Type	Meaning
ATTACK_ROLE_ALL	int	ALL_SYM
ATTACK_ROLE_NONE	int	NONE_SYM
ATTACK_ROLE_GROUND	int	GROUND_SYM
ATTACK_ROLE_AIR	int	AIR_SYM
ATTACK_ROLE_GROUND	int	None of the above.
Calls		
Function	Where Described	
symbols_match	Sec. 2.1.1.5 See Appendix A	

Table 2.6-410: map_role_sym_to_role_number CSU [8.6.9.3.1]

2.6.9.3.2 create_targeting CSU

Parameters		
Parameters	Type	Where Typedef Declared
*table	pointer to DATA UNION	Sec. 2.1.1.5
Calls		
Function	Where Described	
allocate targeting	Sec. 2.9.1.2 See Appendix A	
ft float	Sec. 2.14.1.2.12	
ft symbol	Sec. 2.14.1.2.13	
init target list	Sec. 2.6.9.3.6	
vec init	Sec. 2.6.2.31.1 Vehicles CSCI SDD	
symbols match	Sec. 2.1.1.5 See Appendix A	
map_role_sym_to_role_number	Sec. 2.6.9.3.1	

Table 2.6-411: create_targeting CSU [8.6.9.3.2]

2.6.9.3.3 destroy_targeting CSU

This CSU removes a targeting structure from memory by deallocating the space previously reserved.

Parameters		
Parameters	Type	Where Typedef Declared
*targeting	pointer to TARGETING_VARS	Sec. 2.9.1.2
Calls		
Function	Where Described	
deallocate targeting	Sec. 2.9.1.2 See Appendix A	

Table 2.6-412: destroy_targeting CSU [8.6.9.3.3]

2.6.9.3.4 firestatus_to_string CSU

This CSU converts a firestatus symbolic constant to a character string.

Parameters		
Parameters	Type	Where Typedef Declared
s	unsigned char	Standard

ReturnValues		
Return Value	Type	Meaning
"hold"	pointer to char	FIRESTATUS_HOLD_FIRE
"fire at will"	pointer to char	FIRESTATUS_FIRE_AT_WILL
"fire at designated"	pointer to char	FIRESTATUS_FIRE_AT_DESIGNATED_TARGETS
"fire with leader"	pointer to char	FIRESTATUS_FIRE_AT_WHAT_LEADER_SHOOTS
"fire at position"	pointer to char	FIRESTATUS_FIRE_AT_POSITION
"unknown"	pointer to char	None of the above.

Table 2.6-413: firestatus_to_string CSU [8.6.9.3.4]

2.6.9.3.5 targeting_show CSU

This CSU displays targeting information from a TARGETING_VARS data structure.

Parameters		
Parameters	Type	Where Typedef Declared
*td	pointer to TARGETING_VARS	Sec. 2.9.1.2
flags	int	Standard
Calls		
Function	Where Described	
firestatus_to_string	Sec. 2.6.9.3.4	

Table 2.6-414: targeting_show CSU [8.6.9.3.5]

2.6.9.3.6 init_target_list CSU

This CSU sets the first element of the target list and the target element in a TARGETING_VARS data structure to zero.

Parameters		
Parameters	Type	Where Typedef Declared
*td	pointer to TARGETING_VARS	Sec. 2.9.1.2

Table 2.6-415: init_target_list CSU [8.6.9.3.6]

2.6.9.3.7 set_target_list CSU

Parameters		
Parameters	Type	Where Typedef Declared
*td	pointer to TARGETING_VARS	Sec. 2.9.1.2
*l	unsigned short	Standard

Table 2.6-416: set_target_list CSU [8.6.9.3.7]

2.6.9.3.8 targeting_set_fire_at_pointair CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
x	REAL	sim_types.h
y	REAL	sim_types.h
r	REAL	sim_types.h

Table 2.6-417: targeting_set_fire_at_pointair CSU [8.6.9.3.8]**2.6.9.3.9 targeting_set_hold_fire CSU**

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1

Table 2.6-418: targeting_set_hold_fire CSU [8.6.9.3.9]**2.6.9.3.10 targeting_set_fire_at_will CSU**

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1

Table 2.6-419: targeting_set_fire_at_will CSU [8.6.9.3.10]**2.6.9.3.11 targeting_set_parameters CSU**

Parameters		
Parameters	Type	Where Typedef Declared
*td	pointer to TARGETING_VARS	Sec. 2.9.1.2
firestatus	int	Standard
max_engagement_range	REAL	sim_types.h
marksmanship	REAL	sim_types.h
position_x	REAL	sim_types.h
position_y	REAL	sim_types.h
radius	REAL	sim_types.h
targets[]	unsigned short	Standard

Table 2.6-420: targeting_set_parameters CSU [8.6.9.3.11]

2.6.9.3.12 TARGET_ITEM CSU

This CSU contains a TARGET_ITEM structure definition used by the target code to sort potential targets into a target list. The structure has no tag.

Item	Type	Where Type Defined
ID	unsigned short	Standard
range2	REAL	sim_types.h
priority	int	Standard

Table 2.6-421: TARGET_ITEM Structure Definition

2.6.9.3.13 init_target_items CSU

Parameters		
Parameters	Type	Where Typedef Declared
*targ	pointer to TARGET_ITEM	Sec. 2.6.9.3.12

Table 2.6-422: init_target_items CSU [8.6.9.3.13]

2.6.9.3.14 compare_targets CSU

This CSU is used to sort target priorities. The lower of two target priority numbers has the higher priority.

Parameters		
Parameters	Type	Where Typedef Declared
*item1	pointer to TARGET_ITEM	Sec. 2.6.9.3.12
*item2	pointer to TARGET_ITEM	Sec. 2.6.9.3.12
ReturnValues		
Return Value	Type	Meaning
priority_diff	int	Target priorities used.
(int)(item1->range2 - item2->range2)	int	Distance used.

Table 2.6-423: compare_targets CSU [8.6.9.3.14]

2.6.9.3.15 target_priority CSU

This CSU sets a target's priority for targeting. It is currently used only by attacking air vehicles on their targets.

Parameters		
Parameters	Type	Where Typedef Declared
target object type	ObjectType	p sim.h

ReturnValues		
Return Value	Type	Meaning
TARGET_PRIORITY_ADA	int	Target is antiaircraft.
TARGET_PRIORITY_AIR	int	Target is aircraft.
TARGET_PRIORITY_TANK	int	Target is tank.
TARGET_PRIORITY_DEFAULT	int	Target is none of the above.
Calls		
Function	Where Described	
IS_ANTIAIRCRAFT		
IS_AIRCRAFT		
IS_TANK		

Table 2.6-424: target_priority CSU [8.6.9.3.15]

2.6.9.3.16 target_type_ok CSU

This CSU compares the caller's object type with an enemy and sees if the caller can legally engage him.

Parameters		
Parameters	Type	Where Typedef Declared
attacker_object_type	ObjectType	p_sim.h
targetID	unsigned short	Standard
attack_role	unsigned char	Standard
*targ_priority	pointer to int	Standard
ReturnValues		
Return Value	Type	Meaning
FALSE	int	No target, or no attack role, or target is dead, or target is fast helicopter, or none of the below.
TRUE	int	Attacker is aircraft, or target is ground vehicle, or attack role is air or all.
IS_AIRCRAFT(...)	int	Attacker is antiaircraft.
check_prob(.10)	int	Target is slow helicopter.

Calls	
Function	Where Described
LOOKUP SAFOBJ	Sec. 2.9.1.1 See Appendix A
OBJ OBJECT TYPE	Sec. 2.9.1.1 See Appendix A
is dead	Sec. 2.13.3.1 See Appendix A
OBJ APPEARANCE	Sec. 2.9.1.1 See Appendix A
OBJ VEHICLECLASS	Sec. 2.9.1.1 See Appendix A
IS AIRCRAFT	
target_priority	Sec. 2.6.9.3.15
IS ANTIAIRCRAFT	
IS GROUNDVEH	
IS HELI	
LOOKUP_VELOCITY	Sec. 2.9.1.1 See Appendix A
vec_dot_prod	Sec. 2.6.2.54.1 Vehicles CSCI SDD

Table 2.6-425: target_type_ok CSU [8.6.9.3.16]

2.6.9.3.17 target_in_position CSU

This CSU checks to see if the enemy target is close to a position at which the caller is allowed to shoot.

Parameters		
Parameters	Type	Where Typedef Declared
must be near	int	Standard
*target_pos	pointer to REAL	sim_types.h
*engage_pos	pointer to REAL	sim_types.h
radius2	REAL	sim_types.h
*range2	pointer to REAL	sim_types.h
ReturnValues		
Return Value	Type	Meaning
TRUE	int	must_be_near==FALSE or *range2<radius2.
FALSE	int	must_be_near==TRUE and *range2>=radius2
Calls		
Function	Where Described	
range_squared	Sec. 2.14.3.5.10	

Table 2.6-426: target_in_position CSU [8.6.9.3.17]

2.6.9.3.18 select_weapon_priority_list CSU

This CSU selects a weapon priority list for a given *object_type*.

Parameters		
Parameters	Type	Where Typedef Declared
*td	pointer to TARGETING_VARS	Sec. 2.9.1.2
*wsd	pointer to WEAPON_SYSTEMS_VARS	Sec. 2.9.1.2
object_type	ObjectType	p_sim.h
ReturnValues		
Return Value	Type	Meaning
&wsd->tank_priority_list	pointer to WEAPON_PRIORITY_LIST	Object type is tank.
&wsd->tank_air_list	pointer to WEAPON_PRIORITY_LIST	Object type is aircraft.
&wsd->tank_ground_list	pointer to WEAPON_PRIORITY_LIST	Object type is neither tank nor aircraft.
Calls		
Function	Where Described	
IS_TANK		
IS_AIRCRAFT		

Table 2.6-427: select_weapon_priority_list CSU [8.6.9.3.18]

2.6.9.3.19 update_target_list CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
OBJ_POSITION	Sec. 2.9.1.1 See Appendix A	
OBJ_OBJECT_TYPE	Sec. 2.9.1.1 See Appendix A	
init target list	Sec. 2.6.9.3.6	
init target items	Sec. 2.6.9.3.13	
target type ok	Sec. 2.6.9.3.16	
LOOKUP_POSITION	Sec. 2.9.1.1 See Appendix A	
target in position	Sec. 2.6.9.3.17	
range squared	Sec. 2.14.3.5.10	
qsort		
min	Sec. 2.6.7.3 & Sec. 2.13.3.5 See Appendix A	

Table 2.6-428: update_target_list CSU [8.6.9.3.19]

2.6.9.3.20 choose_target_and_weapon CSU

Parameters		
Parameters	Type	Where Typedef Declared
*td	pointer to TARGETING VARS	Sec. 2.9.1.2
*wsd	pointer to WEAPON SYSTEMS VARS	Sec. 2.9.1.2
*my_position	pointer to REAL	sim_types.h
myID	unsigned int	Standard
Calls		
Function	Where Described	
LOOKUP SAFOBJ	Sec. 2.9.1.1 See Appendix A	
OBJ POSITION	Sec. 2.9.1.1 See Appendix A	
OBJ VELOCITY	Sec. 2.9.1.1 See Appendix A	
OBJ OBJECT TYPE	Sec. 2.9.1.1 See Appendix A	
range squared	Sec. 2.14.3.5.10	
intervis can see pt to pt	Sec. 2.6.5.2.1	
select weapon priority list	Sec. 2.6.9.3.18	
DEBUG TARGETING	Sec. 2.5.2.2 See Appendix A	

Table 2.6-429: choose_target_and_weapon CSU [8.6.9.3.20]

2.6.9.3.21 targeting_tick CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
detection tick	Sec. 2.6.5.1.11	
major detection increase	Sec. 2.6.5.1.13	
major spotter increase	Sec. 2.8.2.3.10	
detection save weight	Sec. 2.6.5.1.12	
spotter save weight	Sec. 2.8.2.3.11	
update target list	Sec. 2.6.9.3.19	
get me a random fraction	Sec. 2.14.3.7.1	
choose target and weapon	Sec. 2.6.9.3.20	
OBJ POSITION	Sec. 2.9.1.1 See Appendix A	
OBJ VEHICLEID	Sec. 2.9.1.1 See Appendix A	
loader tick	Sec. 2.6.9.10.1	
gunner tick	Sec. 2.6.9.5.7	

Table 2.6-430: targeting_tick CSU [8.6.9.3.21]

2.6.9.4 targeting.h CSU

/simnet/src/host/targeting.h

This CSU contains the symbolic constants used by the targeting code.

Constant	Value
MAXIMUM TARGETS TO SORT /* Maximum # of targets to sort */	16
MAXIMUM TARGETS /* Maximum # of targets to consider in a target list */	8
FIRESTATUS HOLD FIRE /* firing status */	0
FIRESTATUS FIRE AT WILL	1
FIRESTATUS FIRE AT POSITION	2
FIRESTATUS FIRE AT WHAT LEADER SHOOTS	3
FIRESTATUS FIRE AT DESIGNATED TARGET	4
FIRESTATUS FIRE AT POINTAIR	5
GUNNER STATE SCANNING /* States that gunner can be in */	0
GUNNER STATE ACQUIRING	1
GUNNER STATE ACQUIRED	2
GUNNER STATE TRACKING	3
GUNNER STATE TRACKED	4
GUNNER STATE FLYING ROUND	5
GUNNER STATE LOST VISIBILITY	6
ACQUIRE RESULT OK /* Results of trying to acquire a target */	0
ACQUIRE RESULT TARGET DEAD	1
ACQUIRE RESULT TARGET GONE	2
TRACK RESULT LOCKED ON /* Results of trying to track a target */	0
TRACK RESULT TRYING	1
TRACK RESULT TARGET DEAD	2
TRACK RESULT TARGET GONE	3
TRACK RESULT TARGET NOT VISIBLE	4
ATTACK ROLE NONE /* Attack roles */	0
ATTACK ROLE ALL	1
ATTACK ROLE GROUND	2
ATTACK ROLE AIR	3
TARGET PRIORITY ADA /* Target priority list, lowest number engaged first */	0
TARGET PRIORITY AIR	1
TARGET PRIORITY TANK	2
TARGET PRIORITY DEFAULT	3

Table 2.6-431: targeting.h Constant Definitions

2.6.9.5 gunner.c CSC

/simnet/src/host/gunner.c

This CSC handles tracking a chosen target, bringing a weapon to bear on this target, firing the weapon, and delivering a round to the target.

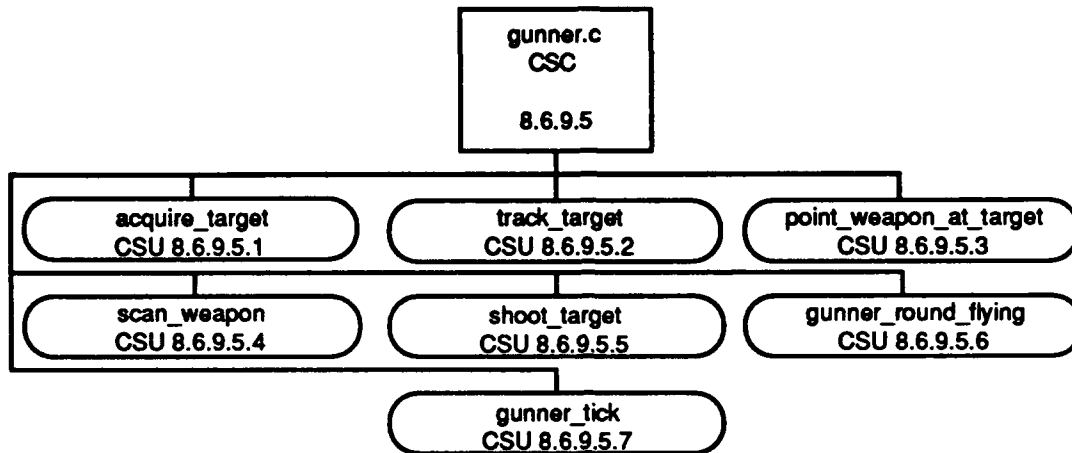


Figure 2.6-38: gunner.c CSC Structure

2.6.9.5.1 acquire_target CSU

This CSU performs target acquisition for a given weapon and target.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*weapon	pointer to WEAPON_VARS	Sec. 2.9.1.2
targetID	unsigned short	Standard
ReturnValues		
Return Value	Type	Meaning
ACQUIRE_RESULT_TARGET_GONE	int	No target.
ACQUIRE_RESULT_TARGET_DEAD	int	Dead target.
ACQUIRE_RESULT_OK	int	Target acquired.
Calls		
Function	Where Described	
LOOKUP_VEHICLE	Sec. 2.9.3.2 See Appendix A	
is_dead	Sec. 2.13.3.1 See Appendix A	
LOOKUP_APPEARANCE	Sec. 2.9.1.1 See Appendix A	
LOOKUP_VEHICLECLASS	Sec. 2.9.1.1 See Appendix A	
point_weapon_at_target	Sec. 2.6.9.5.3	

Table 2.6-432: acquire_target CSU [8.6.9.5.1]

2.6.9.5.2 track_target CSU

This CSU performs target tracking for a given weapon and target.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*weapon	pointer to WEAPON_VARS	Sec. 2.9.1.2
targetID	unsigned short	Standard
ReturnValues		
Return Value	Type	Meaning
TRACK_RESULT_TARGET_GONE	int	No target.
TRACK_RESULT_TARGET_DEAD	int	Dead target.
TRACK_RESULT_TARGET_NOT_VISIBLE	int	Target not visible.
TRACK_RESULT_LOCKED_ON	int	Weapon successfully pointed at target.
TRACK_RESULT_TRYING	int	None of the above.
Calls		
Function	Where Described	
OBJ_VEHICLEID	Sec. 2.9.1.1 See Appendix A	
OBJ_POSITION	Sec. 2.9.1.1 See Appendix A	
LOOKUP_POSITION	Sec. 2.9.1.1 See Appendix A	
LOOKUP_VELOCITY	Sec. 2.9.3.2 See Appendix A	
LOOKUP_VEHICLE	Sec. 2.9.3.2 See Appendix A	
is dead	Sec. 2.13.3.1 See Appendix A	
LOOKUP_APPEARANCE	Sec. 2.9.1.1 See Appendix A	
intervis can see pt to pt	Sec. 2.6.5.2.1	
radiate target	Sec. 2.6.9.8.19	
point weapon at target	Sec. 2.6.9.5.3	

Table 2.6-433: track_target CSU [8.6.9.5.2]

2.6.9.5.3 point_weapon_at_target CSU

This CSU points the weapon at the target.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*weapon	pointer to WEAPON_VARS	Sec. 2.9.1.2
targetID	unsigned short	Standard

ReturnValues		
Return Value	Type	Meaning
turret_point_at_target(...)	int	Turreted weapon.
pilot_point_at_target(...)	int	Air vehicle.
FALSE	int	None of the above.
Calls		
Function	Where Described	
turret_point_at_target	Sec. 2.6.9.6.5	
pilot_point_at_target	Sec. 2.6.4.2.50	

Table 2.6-434: point_weapon_at_target CSU [8.6.9.5.3]

2.6.9.5.4 scan_weapon CSU

This CSU causes the turret to scan.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
*weapon	pointer to WEAPON VARS	Sec. 2.9.1.2
Calls		
Function	Where Described	
turret_scan	Sec. 2.6.9.6.6	

Table 2.6-435: scan_weapon CSU [8.6.9.5.4]

2.6.9.5.5 shoot_target CSU

This CSU attempts to fire a weapon at a target.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
*weapon	pointer to WEAPON VARS	Sec. 2.9.1.2
targetID	unsigned short	Standard
ReturnValues		
Return Value	Type	Meaning
ROUND_NOT_SHOT_BECAUSE_OF_NO_LOS	int	Weapon not fired because of no line-of-sight visibility to the target position.
ROUND_AWAY	int	Weapon fired.
ROUND_NOT_SHOT_BECAUSE_OF_NO_LEAD	int	Weapon not fired because of no line_of_sight visibility to the predicted target position.

Calls	
Function	Where Described
OBJ VEHICLEID	Sec. 2.9.1.1 See Appendix A
OBJ POSITION	Sec. 2.9.1.1 See Appendix A
LOOKUP VELOCITY	Sec. 2.9.3.2 See Appendix A
saf vehicle est position	Sec. 2.6.1.1.58
LOOKUP SAFOBJ	Sec. 2.9.1.1 See Appendix A
intervis can see pt to pt	Sec. 2.6.5.2.1
fire missile at target	Sec. 2.6.9.1.6
range squared	Sec. 2.14.3.5.10
vec copy	Sec. 2.6.2.59.1 Vehicles CSCI SDD
fire weapon at target	Sec. 2.6.9.8.21

Table 2.6-436: shoot_target CSU [8.6.9.5.5]

2.6.9.5.6 gunner_round_flying CSU

This CSU sets the gunner state to GUNNER_STATE_FLYING_ROUND.

Parameters		
Parameters	Type	Where Typedef Declared
*td	pointer to TARGETING_VARS	Sec. 2.9.1.2
ReturnValues		
Return Value	Type	Meaning
td->gunner_state == GUNNER_STATE_FLYING_ROUND	int	Gunner state set.

Table 2.6-437: gunner_round_flying CSU [8.6.9.5.6]

2.6.9.5.7 gunner_tick CSU

This CSU performs the gunner's tasks. The gunner's job is to acquire, track, and shoot the chosen target with the chosen weapon. If he doesn't have a target, he "scans" his weapon.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
OBJ VEHICLEID	Sec. 2.9.1.1 See Appendix A	
DEBUG GUNNER	Sec. 2.5.2.2 See Appendix A	
scan weapon	Sec. 2.6.9.5.4	
acquire target	Sec. 2.6.9.5.1	
track target	Sec. 2.6.9.5.2	
shoot target	Sec. 2.6.9.5.5	
fly_round	Sec. 2.6.9.8.22	

Table 2.6-438: gunner_tick CSU [8.6.9.5.7]

2.6.9.6 turret.c CSC

/simnet/src/host/turret.c

This CSC contains the code dealing with turret issues, including slewing, scanning, target tracking, and taking damage.

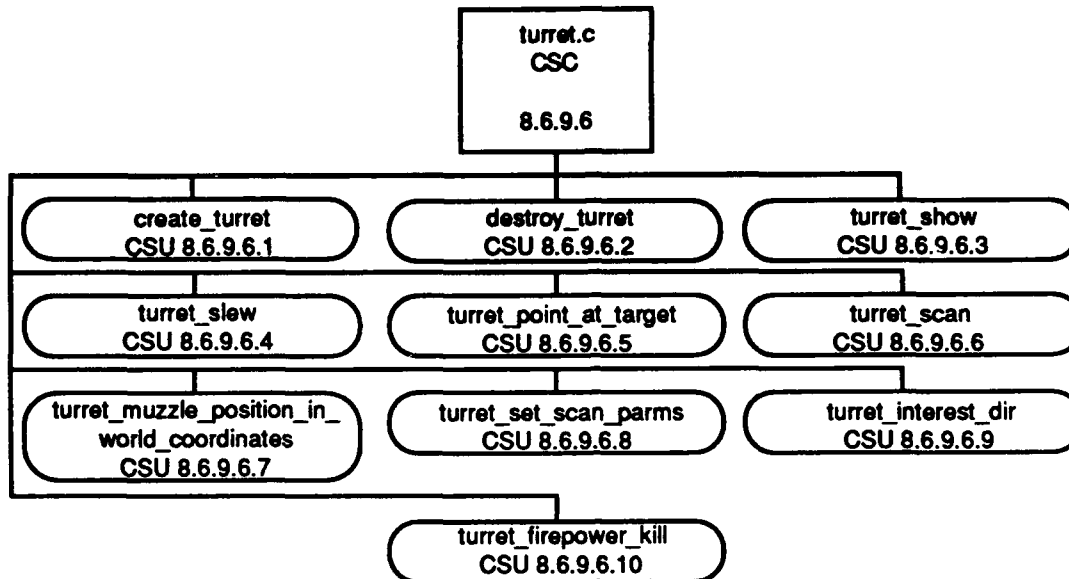


Figure 2.6-39: turret.c.c CSC Structure

2.6.9.6.1 create_turret CSU

This CSU allocates memory space for a TURRET_VARS data structure and initializes its members.

Parameters		
Parameters	Type	Where Typedef Declared
*table	pointer to DATA_UNION	Sec. 2.1.1.5
ReturnValues		
Return Value	Type	Meaning
tur	pointer to TURRET_VARS	Turret created.
Calls		
Function	Where Described	
allocate turret	Sec. 2.9.1.2 See Appendix A	
deg to rad	sim macros.h	
ft float	Sec. 2.14.1.2.12	

Table 2.6-439: create_turret CSU [8.6.9.6.1]

2.6.9.6.2 destroy_turret CSU

This CSU uses `deallocate_turret` to deallocate memory space previously reserved for a `TURRET_VARS` data structure.

Parameters		
Parameters	Type	Where Typedef Declared
*tur	pointer to TURRET_VARS	Sec. 2.9.1.2
Calls		
Function	Where Described	
deallocate_turret	Sec. 2.9.1.2 See Appendix A	

Table 2.6-440: destroy_turret CSU [8.6.9.6.2]

2.6.9.6.3 turret_show CSU

This CSU displays turret information from a `TURRET_VARS` data structure.

Parameters		
Parameters	Type	Where Typedef Declared
*tur	pointer to TURRET_VARS	Sec. 2.9.1.2
flags	int	Standard
Calls		
Function	Where Described	
rad to deg	sim macros.h	

Table 2.6-441: turret_show CSU [8.6.9.6.3]

2.6.9.6.4 turret_slew CSU

This CSU moves the turret to a requested azimuth and elevation.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
azimuth rps	REAL	sim types.h
elevation rps	REAL	sim types.h
ReturnValues		
Return Value	Type	Meaning
azimuth_satisfied && elevation_satisfied	int	1 if succeeded, 0 if failed to reach both azimuth and elevation.

Calls	
Function	Where Described
OBJ_TIME_SINCE_LAST_TICK	Sec. 2.9.1.1 See Appendix A
angle clip	Sec. 2.14.3.5.4
sign	
abs	Sec. 2.6.7.3 & Sec. 2.13.3.2 See Appendix A
RANGE_CLIP	Sec. 2.14.3.9 See Appendix A
DEBUG_TURRET	Sec. 2.5.2.2 See Appendix A
OBJ_VEHICLEID	Sec. 2.9.1.1 See Appendix A
OBJ_TURRETAZIMUTH	Sec. 2.9.1.1 See Appendix A
radians to simnet angle	libapp.h
clip angle positive	Sec. 2.14.3.5.5
OBJ_GUNELEVATION	Sec. 2.9.1.1 See Appendix A

Table 2.6-442: turret_slew CSU [8.6.9.6.4]

2.6.9.6.5 turret_point_at_target CSU

This CSU calculates the desired azimuth and elevation and rates of travel to point the turret at the target.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
targetID	unsigned short	Standard
ReturnValues		
Return Value	Type	Meaning
FALSE	int	Turret trashed.
turret_slew(...)	int	Normal end.
Calls		
Function	Where Described	
vec sub	Sec. 2.6.2.35.1 Vehicles CSCI SDD	
LOOKUP_POSITION	Sec. 2.9.1.1 See Appendix A	
OBJ_POSITION	Sec. 2.9.1.1 See Appendix A	
mat vec mul	Sec. 2.6.2.59.1 Vehicles CSCI SDD	
OBJ_HULL_TO_WORLD	Sec. 2.9.1.1 See Appendix A	
vec_mag3	sim_macros.h	
s_atan2	Sec. 2.14.3.9 See Appendix A	
angle clip	Sec. 2.14.3.5.4	
abs	Sec. 2.6.7.3 & Sec. 2.13.3.2 See Appendix A	
turret_slew	Sec. 2.6.9.6.4	

Table 2.6-443: turret_point_at_target CSU [8.6.9.6.5]

2.6.9.6.6 turret_scan CSU

This CSU performs the calculations for scanning the turret.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
angle clip	Sec. 2.14.3.5.4	
get me a random fraction	Sec. 2.14.3.7.1	
DEBUG TURRET	Sec. 2.5.2.2 See Appendix A	
OBJ VEHICLEID	Sec. 2.9.1.1 See Appendix A	
rad to deg	sim macros.h	
turret slew	Sec. 2.6.9.6.4	

Table 2.6-444: turret_scan CSU [8.6.9.6.6]

2.6.9.6.7 turret_muzzle_position_in_world_coordinates CSU

Parameters		
Parameters	Type	Where Typedef Declared
*tur	pointer to TURRET_VARS	
*pos	pointer to REAL	sim_types.h
**htw	pointer to pointer to REAL	sim_types.h
*muzzle_vec	pointer to REAL	sim_types.h
Calls		
Function	Where Described	
vec mat mul	Sec. 2.6.2.56.1 Vehicles CSCI SDD	
vec_add	Sec. 2.6.2.57.1 Vehicles CSCI SDD	

Table 2.6-445: turret_muzzle_position_in_world_coordinates CSU [8.6.9.6.7]

2.6.9.6.8 turret_set_scan_parms CSU

This CSU sets scan parameters.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
left	REAL	sim_types.h
right	REAL	sim_types.h

Calls	
Function	Where Described
IS ANTIAIRCRAFT	
OBJ_OBJECT_TYPE	Sec. 2.9.1.1 See Appendix A
deg to rad	sim_macros.h

Table 2.6-446: turret_set_scan_parms CSU [8.6.9.6.8]

2.6.9.6.9 turret_interest_dir CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*vec	pointer to REAL	sim_types.h
Calls		
Function	Where Described	
vec mat mul	Sec. 2.6.2.56.1 Vehicles CSCI SDD	
OBJ_HULL_TO_WORLD	Sec. 2.9.1.1 See Appendix A	

Table 2.6-447: turret_interest_dir CSU [8.6.9.6.9]

2.6.9.6.10 turret_firepower_kill CSU

This CSU kills turret firepower by setting turret_trashed to TRUE.

Parameters		
Parameters	Type	Where Typedef Declared
*tur	pointer to TURRET_VARS	Sec. 2.9.1.2

Table 2.6-448: turret_firepower_kill CSU [8.6.9.6.10]

2.6.9.7 turret.h CSU

/simnet/src/host/turret.h

This CSU contains the symbolic constants used by the turret code.

Constant	Value
MAX_SCAN_IDLE_TIME	30000
MIN_SCAN_IDLE_TIME	10000
SCAN_STATE_SCANNING	0
SCAN_STATE_WAITING	1

Table 2.6-449: turret.h Constant Definitions

2.6.9.8 weapons.c CSC

/simnet/src/host/weapons.c

This CSU contains the code which controls, selects, and de-selects any particular weapon. It also handles the actual tracking of a round to target, firing the weapon, and the round impact and hit upon arrival.

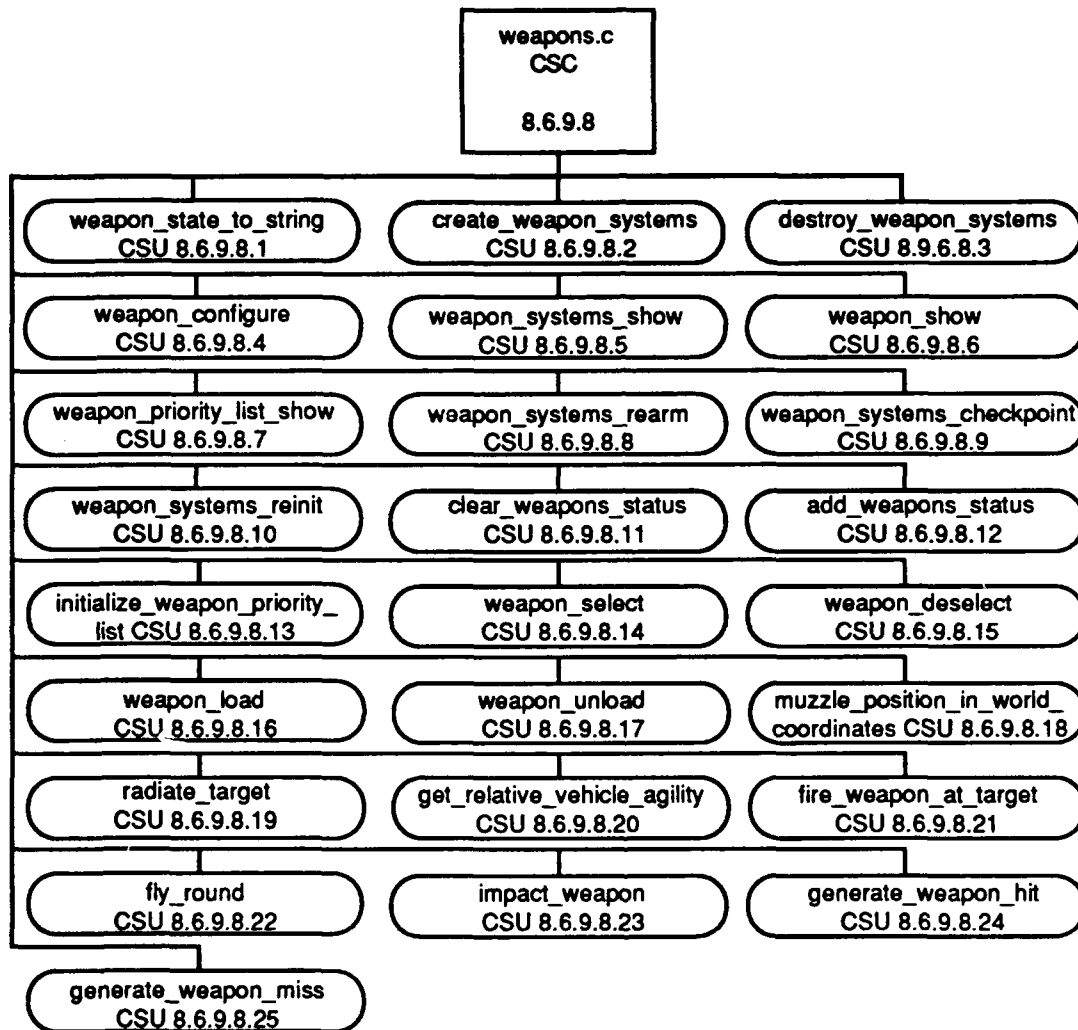


Figure 2.6-40: weapons.c CSC Structure

2.6.9.8.1 weapon_state_to_string CSU

This CSU converts a weapon state variable to a character string.

Parameters		
Parameters	Type	Where Typedef Declared
state	int	Standard

ReturnValues		
Return Value	Type	Meaning
"not selected"	pointer to char	WEAPON_STATE_NOT_SELECTED
"selecting"	pointer to char	WEAPON_STATE_SELECTING
"selected"	pointer to char	WEAPON_STATE_SELECTED
"loading"	pointer to char	WEAPON_STATE_LOADING
"loaded"	pointer to char	WEAPON_STATE_LOADED
"unloading"	pointer to char	WEAPON_STATE_UNLOADING
"deselecting"	pointer to char	WEAPON_STATE_DESELECTING
"unknown"	pointer to char	None of the above.

Table 2.6-450: weapon_state_to_string CSU [8.6.9.8.1]

2.6.9.8.2 create_weapon_systems CSU

This CSU allocates memory space for a WEAPON_SYSTEMS_VARS data structure, configures the weapons, initializes the default weapon, and initializes the tank, air, and ground weapon priority lists.

Parameters		
Parameters	Type	Where Typedef Declared
*table	pointer to DATA UNION	Sec. 2.1.1.5
ownerID	unsigned short	Standard
forceID	ForceID	basic.h
percent_amm0	REAL	sim_types.h
ReturnValues		
Return Value	Type	Meaning
NULL	pointer to WEAPON_SYSTEMS_VARS	Unable to configure a weapon.
wsd	pointer to WEAPON_SYSTEMS_VARS	Created data structure.
Calls		
Function	Where Described	
allocate_weapon_systems	Sec. 2.9.1.2 See Appendix A	
ft_table	Sec. 2.14.1.2.14	
ft_symbol	Sec. 2.14.1.2.13	
find_tag	Sec. 2.1.1.4.3	
DEBUG_WEAPON	Sec. 2.5.2.2 See Appendix A	
weapon_configure	Sec. 2.6.9.8.4	
symbols_match	Sec. 2.1.1.5 See Appendix A	
initialize_weapon_priority_list	Sec. 2.6.9.8.13	

Table 2.6-451: create_weapon_systems CSU [8.6.9.8.2]

2.6.9.8.3 destroy_weapon_systems CSU

This CSU deallocates memory space previously reserved for a WEAPON_SYSTEMS_VARS data structure..

Parameters		
Parameters	Type	Where Typedef Declared
*wsd	pointer to WEAPON_SYSTEMS_VARS	Sec. 2.9.1.2
Calls		
Function	Where Described	
destroy_missile	Sec. 2.6.9.1.4	
deallocate_weapon_systems	Sec. 2.9.1.2 See Appendix A	

Table 2.6-452: destroy_weapon_systems CSU [8.6.9.8.3]

2.6.9.8.4 weapon_configure CSU

This CSU initializes the weapon configuration.

Parameters		
Parameters	Type	Where Typedef Declared
*weapon	pointer to WEAPON_VARS	Sec. 2.9.1.2
*table	pointer to DATA_UNION	Sec. 2.1.1.5
ownerID	unsigned short	Standard
forceID	ForceID	basic.h
percent_ammo	REAL	sim_types.h
Calls		
Function	Where Described	
ft symbol	Sec. 2.14.1.2.13	
find tag	Sec. 2.1.1.4.3	
ft int	Sec. 2.14.1.2.11	
ft float	Sec. 2.14.1.2.12	
create_missile	Sec. 2.6.9.1.3	

Table 2.6-453: weapon_configure CSU [8.6.9.8.4]

2.6.9.8.5 weapon_systems_show CSU

This CSU displays weapon systems information from a WEAPON_SYSTEMS_VARS data structure.

Parameters		
Parameters	Type	Where Typedef Declared
*wsd	pointer to WEAPON_SYSTEMS_VARS	Sec. 2.9.1.2
flags	int	Standard

Calls	
Function	Where Described
weapon show	Sec. 2.6.9.8.6
weapon_priority_list_show	Sec. 2.6.9.8.7

Table 2.6-454: weapon_systems_show CSU [8.6.9.8.5]

2.6.9.8.6 weapon_show CSU

This CSU displays weapon configuration information from a WEAPON_VARS data structure.

Parameters		
Parameters	Type	Where Typedef Declared
*weapon	pointer to WEAPON_VARS	Sec. 2.9.1.2

Calls	
Function	Where Described
weapon state to string	Sec. 2.6.9.8.1
missile show	Sec. 2.6.9.1.5

Table 2.6-455: weapon_show CSU [8.6.9.8.6]

2.6.9.8.7 weapon_priority_list_show CSU

This CSU displays the selected weapon priority list.

Parameters		
Parameters	Type	Where Typedef Declared
*wsd	pointer to WEAPON_SYSTEMS_VARS	Sec. 2.9.1.2
*wpl	pointer to WEAPON_PRIORITY_LIST	Sec. 2.9.1.2

Table 2.6-456: weapon_priority_list_show CSU [8.6.9.8.7]

2.6.9.8.8 weapon_systems_rearm CSU

This CSU rearms the selected weapon systems

Parameters		
Parameters	Type	Where Typedef Declared
*wsd	pointer to WEAPON_SYSTEMS_VARS	Sec. 2.9.1.2

Table 2.6-457: weapon_systems_rearm CSU [8.6.9.8.8]

2.6.9.8.13 initialize_weapon_priority_list CSU

This CSU initializes a weapon priority list.

Parameters		
Parameters	Type	Where Typedef Declared
*wsd	pointer to WEAPON SYSTEMS VARS	Sec. 2.9.1.2
*wpl	pointer to WEAPON PRIORITY LIST	Sec. 2.9.1.2
*table	pointer to DATA UNION	Sec. 2.1.1.5
Errors		
Error Name	Reason for Error	
"Couldn't find weapon ... for priority list"	Unable to find a weapon for the priority list.	
Calls		
Function	Where Described	
symbols_match	Sec. 2.1.1.5 See Appendix A	
ERROR_OUT	Sec. 2.5.2.2 See Appendix A	

Table 2.6-462: initialize_weapon_priority_list CSU [8.6.9.8.13]

2.6.9.8.14 weapon_select CSU

This CSU selects a weapon on a vehicle.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
*weapon	pointer to WEAPON_VARS	Sec. 2.9.1.2
Calls		
Function	Where Described	
symbols_match	Sec. 2.1.1.5 See Appendix A	
DEBUG_WEAPON	Sec. 2.5.2.2 See Appendix A	

Table 2.6-463: weapon_selectweapon_select CSU [8.6.9.8.14]

2.6.9.8.15 weapon_deselect CSU

This CSU deselects a weapon on a vehicle.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
*weapon	pointer to WEAPON_VARS	Sec. 2.9.1.2

2.6.9.8.9 weapon_systems_checkpoint CSU

Parameters		
Parameters	Type	Where Typedef Declared
*wsd	pointer to WEAPON SYSTEMS VARS	Sec. 2.9.1.2
loads[]	short	Standard

Table 2.6-458: weapon_systems_checkpoint CSU [8.6.9.8.9]**2.6.9.8.10 weapon_systems_reinit CSU**

Parameters		
Parameters	Type	Where Typedef Declared
*wsd	pointer to WEAPON SYSTEMS VARS	Sec. 2.9.1.2
loads[]	short	Standard

Table 2.6-459: weapon_systems_reinit CSU [8.6.9.8.10]**2.6.9.8.11 clear_weapons_status CSU**

This CSU zeroes weapons status data.

Parameters		
Parameters	Type	Where Typedef Declared
*data	pointer to WEAPONS STATUS DATA	Sec. 2.9.1.2

Table 2.6-460: clear_weapons_status CSU [8.6.9.8.11]**2.6.9.8.12 add_weapons_status CSU**

This CSU updates weapons status data.

Parameters		
Parameters	Type	Where Typedef Declared
*wsd	pointer to WEAPON SYSTEMS VARS	Sec. 2.9.1.2
*data	pointer to WEAPONS STATUS DATA	Sec. 2.9.1.2
Calls		
Function	Where Described	
symbols_match	Sec. 2.1.1.5 See Appendix A	

Table 2.6-461: add_weapons_status CSU [8.6.9.8.12]

Calls	
Function	Where Described
symbols_match	Sec. 2.1.1.5 See Appendix A
DEBUG_WEAPON	Sec. 2.5.2.2 See Appendix A

Table 2.6-464: weapon_deselect CSU [8.6.9.8.15]

2.6.9.8.16 weapon_load CSU

This CSU loads a weapon that is mounted on a vehicle.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
*weapon	pointer to WEAPON VARS	Sec. 2.9.1.2
Calls		
Function	Where Described	
DEBUG_WEAPON	Sec. 2.5.2.2 See Appendix A	

Table 2.6-465: weapon_load CSU [8.6.9.8.16]

2.6.9.8.17 weapon_unload CSU

This CSU unloads a weapon that is mounted on a vehicle.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
*weapon	pointer to WEAPON VARS	Sec. 2.9.1.2
Calls		
Function	Where Described	
DEBUG_WEAPON	Sec. 2.5.2.2 See Appendix A	

Table 2.6-466: weapon_unload CSU [8.6.9.8.17]

2.6.9.8.18 muzzle_position_in_world_coordinates CSU

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
*weapon	pointer to WEAPON VARS	Sec. 2.9.1.2
*muzzle_vec	pointer to REAL	sim types.h

Calls	
Function	Where Described
OBJ POSITION	Sec. 2.9.1.1 See Appendix A
OBJ HULL TO WORLD	Sec. 2.9.1.1 See Appendix A
turret_muzzle_position_in_world_coordinates	Sec. 2.6.9.6.7
vec mat mul	Sec. 2.6.2.56.1 Vehicles CSCI SDD
vec add	Sec. 2.6.2.57.1 Vehicles CSCI SDD

Table 2.6-467: muzzle_position_in_world_coordinates CSU [8.6.9.8.18]

2.6.9.8.19 radiate_target CSU

This CSU radiates the target with radar, after verifying that it is necessary.

Parameters		
Parameters	Type	Where Typedef Declared
attackerID	unsigned short	Standard
*attacker_position	pointer to REAL	sim_types.h
targetID	unsigned short	Standard
*weapon	pointer to WEAPON_VARS	Sec. 2.9.1.2
Calls		
Function	Where Described	
simnet_send_radiate		

Table 2.6-468: radiate_target CSU [8.6.9.8.19]

2.6.9.8.20 get_relative_vehicle_agility CSU

This CSU returns a probability-of-hit modifier for non-guided rounds.

Parameters		
Parameters	Type	Where Typedef Declared
my_object_type	ObjectType	p_sim.h
his_object_type	ObjectType	p_sim.h
*his_velocity	pointer to REAL	sim_types.h

ReturnValues		
Return Value	Type	Meaning
1.0	REAL	Shooter is anti-aircraft; or shooter is aircraft; or target is ground vehicle; or shooter is ground vehicle and target is not an air vehicle; or none of the below.
0.01	REAL	Shooter is ground vehicle and target is plane.
0.9	REAL	Shooter is ground vehicle and target is slow helicopter.
0.1	REAL	Shooter is ground vehicle and target is fast helicopter.
Calls		
Function	Where Described	
IS ANTIAIRCRAFT		
IS AIRCRAFT		
IS GROUNDVEH		
IS PLANE		
IS HELI		
VEC SMALLER3	Sec. 2.14.3.9 See Appendix A	

Table 2.6-469: get_relative_vehicle_agility CSU [8.6.9.8.20]

2.6.9.8.21 fire_weapon_at_target CSU

This CSU fires a weapon at a target. It calculates whether the target is hit or not, modifying the probability of hit according to target agility and shooter marksmanship.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*weapon	pointer to WEAPON_VARS	Sec. 2.9.1.2
attackerID	unsigned short	Standard
*attacker_pos	pointer to REAL	sim_types.h
targetID	unsigned short	Standard
*target_pos	pointer to REAL	sim_types.h
*target_vel	pointer to REAL	sim_types.h
flyout time in sec	REAL	sim_types.h
range	REAL	sim_types.h
*predicted_position	pointer to REAL	sim_types.h

Calls	
Function	Where Described
OBJ OBJECT TYPE	Sec. 2.9.1.1 See Appendix A
LOOKUP OBJECT TYPE	Sec. 2.9.1.1 See Appendix A
LOOKUP VELOCITY	Sec. 2.9.1.1 See Appendix A
saf_vehicle_next_event_id	Sec. 2.6.1.1.28
muzzle_position_in_world_coordinates	Sec. 2.6.9.8.18
vec sub	Sec. 2.6.2.65.1 Vehicles CSCI SDD
vec normalize	Sec. 2.6.2.63.1 Vehicles CSCI SDD
vec scale	Sec. 2.6.2.64.1 Vehicles CSCI SDD
simnet_send_fire	
vec copy	Sec. 2.6.2.59.1 Vehicles CSCI SDD
intervis_get_view	Sec. 2.6.5.2.2
database_hitmodel_query	Sec. 2.6.8.9 See Appendix A
get_relative_vehicle_agility	Sec. 2.6.9.8.20
check_prob	Sec. 2.14.3.7.2
DEBUG WEAPON	Sec. 2.5.2.2 See Appendix A

Table 2.6-470: fire_weapon_at_target CSU [8.6.9.8.21]

2.6.9.8.22 fly_round CSU

This CSU does the calculations for a round or a missile in flight.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*weapon	pointer to WEAPON_VARS	Sec. 2.9.1.2
targetID	unsigned short	Standard
ReturnValues		
Return Value	Type	Meaning
ROUND_DONE_FLYING	int	Burst occurred with conventional round.
ROUND_IN_FLIGHT	int	No burst occurred with conventional round.
fly_missile(...)	int	It's a missile.
Calls		
Function	Where Described	
impact_weapon	Sec. 2.6.9.8.23	
track_target	Sec. 2.6.9.5.2	
DEBUG WEAPON	Sec. 2.5.2.2 See Appendix A	
fly_missile	Sec. 2.6.9.1.11	

Table 2.6-471: fly_round CSU [8.6.9.8.22]

2.6.9.8.23 impact_weapon CSU

This CSU decides if the weapon hit or missed an air vehicle.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
*weapon	pointer to WEAPON VARS	Sec. 2.9.1.2
targetID	unsigned short	Standard
Calls		
Function	Where Described	
LOOKUP POSITION	Sec. 2.9.1.1 See Appendix A	
DEBUG WEAPON	Sec. 2.5.2.2 See Appendix A	
generate weapon miss	Sec. 2.6.9.8.25	
generate_weapon_hit	Sec. 2.6.9.8.24	

Table 2.6-472: impact_weapon CSU [8.6.9.8.23]

2.6.9.8.24 generate_weapon_hit CSU

This CSU generates a weapon hit.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF OBJECT	Sec. 2.9.1.1
*weapon	pointer to WEAPON VARS	Sec. 2.9.1.2
targetID	unsigned short	Standard
Calls		
Function	Where Described	
OBJ VEHICLEID	Sec. 2.9.1.1 See Appendix A	
OBJ POSITION	Sec. 2.9.1.1 See Appendix A	
LOOKUP POSITION	Sec. 2.9.1.1 See Appendix A	
LOOKUP_HULL_TO_WORLD	Sec. 2.9.1.1 See Appendix A	
LOOKUP_VEHICLECLASS	Sec. 2.9.1.1 See Appendix A	
LOOKUP_TURRETAZIMUTH	Sec. 2.9.1.1 See Appendix A	
range squared	Sec. 2.14.3.5.10	
get impact and trajectory	Sec. 2.6.4.5.1	
simnet send impact		
MOMENTUM	Sec. 2.14.3.9 See Appendix A	
ENERGY	Sec. 2.14.3.9 See Appendix A	
DEBUG WEAPON	Sec. 2.5.2.2 See Appendix A	

Table 2.6-473: generate_weapon_hit CSU [8.6.9.8.24]

2.6.9.8.25 generate_weapon_miss CSU

This CSU generates a weapon miss. The symbolic constant MISS_FACTOR is defined before this CSU (#define MISS_FACTOR 10.0).

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
*weapon	pointer to WEAPON_VARS	Sec. 2.9.1.2
targetID	unsigned short	Standard
Calls		
Function	Where Described	
OBJ_VEHICLEID	Sec. 2.9.1.1 See Appendix A	
OBJ_POSITION	Sec. 2.9.1.1 See Appendix A	
LOOKUP_POSITION	Sec. 2.9.1.1 See Appendix A	
vec_copy	Sec. 2.6.2.59.1 Vehicles CSCI SDD	
get_me_a_random_fraction	Sec. 2.14.3.7.1	
range_squared	Sec. 2.14.3.5.10	
simnet_send_impact		
MOMENTUM	Sec. 2.14.3.9 See Appendix A	
ENERGY	Sec. 2.14.3.9 See Appendix A	
DEBUG_WEAPON	Sec. 2.5.2.2 See Appendix A	

Table 2.6-474: generate_weapon_miss CSU [8.6.9.8.25]

2.6.9.9 weapons.h CSU

/simnet/src/host/weapons.h

This CSU contains the structure definition and symbolic constants used by the weapons code.

Constant	Value
MAX_WEAPONS	4
WEAPON_STATE_NOT_SELECTED	0
WEAPON_STATE_SELECTING	1
WEAPON_STATE_SELECTED	2
WEAPON_STATE_LOADING	3
WEAPON_STATE_LOADED	4
WEAPON_STATE_UNLOADING	5
WEAPON_STATE_DESELECTING	6
ROUND_AWAY	0
ROUND_NOT_SHOT_BECAUSE_OF_NO_LOS	1
ROUND_NOT_SHOT_BECAUSE_OF_NO_LEAD	2
ROUND_IN_FLIGHT	3
ROUND_DONE_FLYING	4
MAX_WEAPONS_IN_REPORT	(10 * MAX_WEAPONS)

Table 2.6-475: weapons.h Constant Definitions

The following typedef struct is tagged `gun_burst_description`.

Item	Type	Where Type Defined
burst event id	int	Standard
targetID	short	Standard
range	REAL	sim_types.h
firing position	VECTOR	sim_types.h
target predicted position	VECTOR	sim_types.h
impact time	unsigned int	Standard
is going to hit	int	Standard

Table 2.6-476: GUN_BURST_DESCRIPTION Structure Definition

The following typedef struct is tagged `weapon_status_data`.

Item	Type	Where Type Defined
num weapons	int	Standard
*weapon names[MAX WEAPONS IN REPORT]	pointer to char	Standard
rounds available[MAX WEAPONS IN REPORT]	int	Standard
max rounds[MAX WEAPONS IN REPORT]	int	Standard

Table 2.6-477: WEAPON_STATUS_DATA Structure Definition

2.6.9.10 loader.c CSC

/simnet/src/host/loader.c

This CCSC does the selection of weapon to use and the loading and unloading of rounds. Transition times between rounds and weapon selections are taken into account by this code.

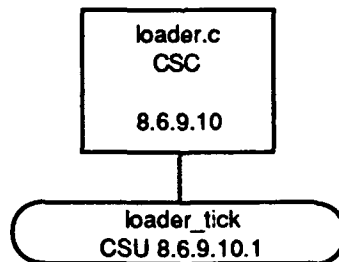


Figure 2.6-41: loader.c CSC Structure

2.6.9.10.1 loader_tick CSU

This CSU selects and loads the chosen weapon. If the commander changes the chosen weapon, the old weapon may have to be unloaded or unselected.

Parameters		
Parameters	Type	Where Typedef Declared
*safobj	pointer to SAF_OBJECT	Sec. 2.9.1.1
Calls		
Function	Where Described	
weapon_select	Sec. 2.6.9.8.14	
weapon_deselect	Sec. 2.6.9.8.15	
OBJ_VEHICLEID	Sec. 2.9.1.1 See Appendix A	
weapon_load	Sec. 2.6.9.8.16	
weapon_unload	Sec. 2.6.9.8.17	

Table 2.6-478: loader_tick CSU [8.6.9.10.1]

2.7 REMOTE VEHICLES CSC

Like the local vehicles, remote vehicles are also ticked by the scheduler. The primary difference is that the local vehicles are simulated by the simhost and the remote vehicles are not. The function of the remote vehicles code is to maintain an internal representation of the state of vehicles which are simulated by other computers on the SIMNET LAN. These computers can be other SAF computers as well as manned vehicle simulators or MCC vehicles.

Remote vehicles broadcast vehicle appearance packets (VAP) on the network whenever their state changes beyond certain thresholds. The SIMNET interface receives these packets and queues them on each remote vehicle. When the remote vehicle is ticked, it updates its state to that described in the VAP. In some cases, the remote vehicle code will extrapolate the position of the remote vehicle at the present time from that received in the last packet. By representing the remote vehicles with the same internal interface as the local vehicles, it is not necessary for a SAF vehicle doing targeting to distinguish between local and remote vehicles.

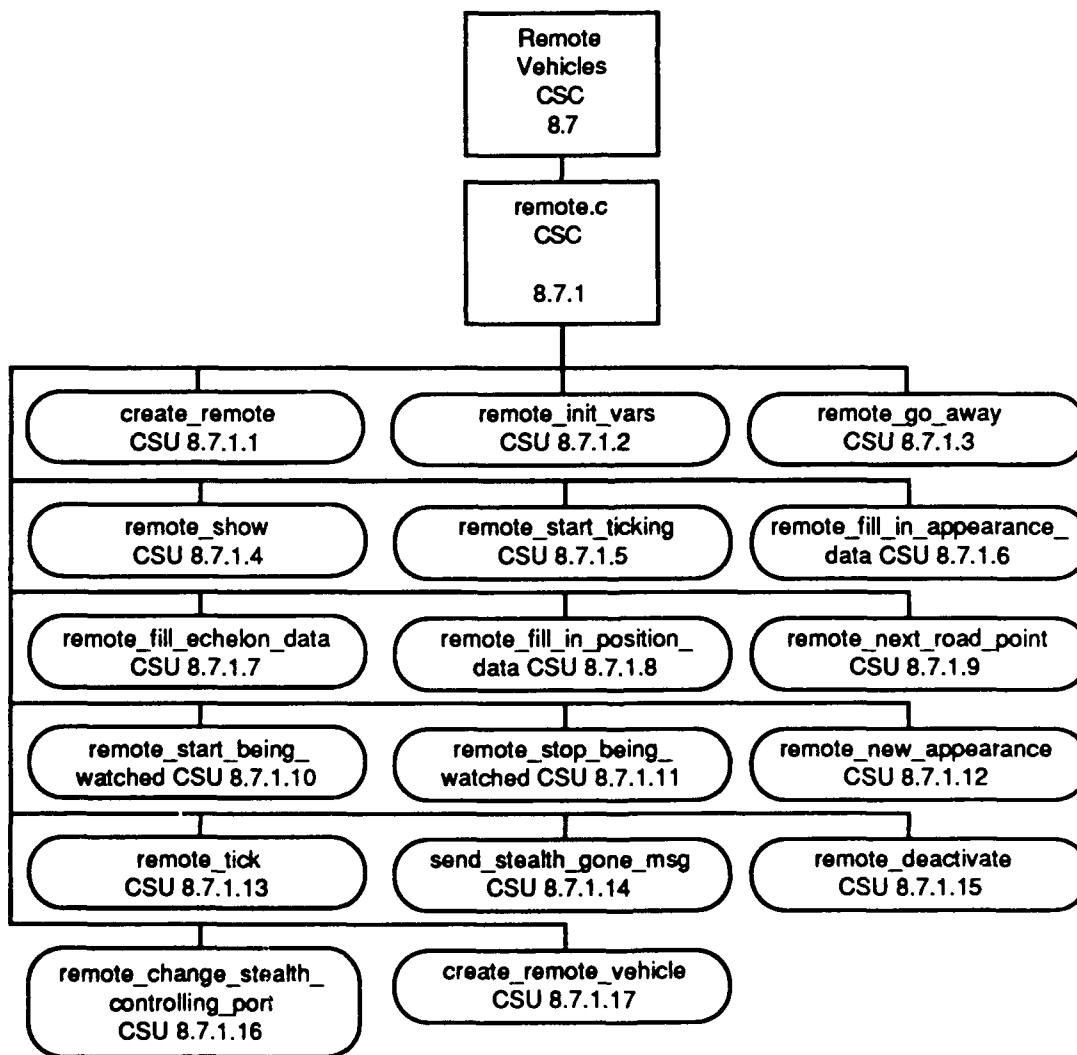


Figure 2.7-1: Remote Vehicles CSC Structure